Impact of mobility and deployment in confined spaces on low power and lossy network

Jinpeng Wang

UNIVERSITÉ CLERMONT AUVERGNE

ÉCOLE DOCTORALE
SCIENCES POUR L'INGÉNIEUR DE CLERMONT-FERRAND

Thése

Présentée par

Jinpeng WANG

Pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Spécialité : Informatique

Titre

Impact of mobility and deployment in confined spaces on low power and lossy network

Soutenue publiquement le 02 Juillet 2019, devant le jury :

Rapporteurs:	Mme. Marion BERBINEAU	Directeur de recherche à IFSTTAR
	M. Adrien VAN DEN BOSSCHE	MCF-HDR à l'Université Toulouse-Jean Jaurés
Examinateurs:	Mme. Pascale MINET	Chargée de recherche HDR à Inria Paris
	M. Alexandre GUITTON	Professeur à l'Université Clermont Auvergne
Directeur de thèse:	M. Michel Misson	Professeur à l'Université Clermont Auvergne
Co-encadrant:	M. Gérard Chalhoub	MCF-HDR à l'Université Clermont Auvergne

UNIVERSITÉ CLERMONT AUVERGNE

DOCTORAL THESIS

Impact of mobility and deployment in confined spaces on low power and lossy network

Author:
Jinpeng WANG

Supervisor:
Pr. Michel MISSON
Dr. Gérard CHALHOUB

A thesis submitted in fulfillment of the requirements for the degree of PhD in Computer Science

in the

LIMOS UMR 6158 CNRS Laboratory
Doctoral school of science for engineering

July 9, 2019

Acknowledgements

Road to a PhD is not always smooth. Many people have assisted and supported me. I am glad to have such an opportunity to express my great gratitude and respect to people who helped me when I was pursuing my PhD. Without their supports I cannot go so far.

First of all, I would like to express my deep appreciation to my thesis director Prof. Michel MISSON and my PhD advisor my Dr. Gérard CHALHOUB, for their warm encouragement and thoughtful support during this research. Their guidance and experience taught me a lot about conducting good research. Their advice on both research as well as on my career have been priceless. Thank you a lot for you both.

I would like to thank Prof. Marion BERBINEAU and Dr. Adrien VAN DEN BOSSCHE for accepting to review my manuscript. Thank you for your availability and your relevant and constructive comments. Their constructive feedback and suggestions greatly improved this thesis.

I also want to thank Dr. Pascale MINET and Prof. Alexandre GUITTON for accepting to be part of my jury. Thank you for your availability.

I would like to thank European Regional Development Fund (FEDER) program of 2014-2020, the region council of Auvergne-Rhône-Alpes and the Digital Trust Chair of the Clermont Auvergne University for the financial support of my thesis.

A special thanks to all my colleagues from C6 office. It was a great pleasure to work with my fellow PhD students: Hamadoun, Mouna, Sylvestre, David, Honoré, Ali, Thérèse and Moussa.

I also want to thank to my former master advisor, Prof. Alain Quilliot. Thanks for your scientific advice and knowledge and insightful discussions and suggestions.

Finally, and most significantly, a greatest gratitude to my parents. I am grateful to all of you, for your encouragements and love.

Abstract

Wireless Sensor Networks (WSNs) technology is one of the building blocks of the Internet of Things (IoT). Due to their features of easy deployment and flexibility, they are used in many application domains. Low-Power and Lossy Networks (LLNs) are a special type of WSNs in which nodes are largely resources constrained. For LLNs, convergecast is one of the basic traffic modes, where all traffic in the network is destined to a predefined destination called the sink. While considering the IoT application domains, convergecast is not the only traffic mode in the network. The sink needs to send commands to certain sensors to perform actions. In this application, anycast is another basic traffic mode. In anycast, the traffic from the sink is destined to any member of a group of potential receivers in the network.

Traditionally LLNs are formed by static sensor nodes and rarely change positions. Due to the strict resource constraints in computation, energy and memory of LLNs, most routing protocols only support static network. However, mobility has become an important requirement for many emerging applications. In these applications, certain nodes are free to move and organize themselves into a connected network. The topology would continuously change due to the movement of nodes and radio links instability. This is a hard task for most routing protocols of LLNs to adapt rapidly to the movement and to reconstruct topology in a timely manner.

The goal of this thesis is to propose an efficient mobility support for routing protocols in LLNs. We focus on convergecast and anycast, which are the most used traffic modes in LLNs, in mobile network scenarios.

We propose an enhancement mechanism, named RL (RSSI and Level), to support routing protocols in convergecast LLNs in mobility. This mechanism helps routing protocol make faster decisions for detecting mobility and updating next-hop neighbors but suffers from high overhead. We propose a dynamic control message management to enhance the overhead performance of RL and implement it on top of Routing Protocol for Low-power and Lossy network (RPL) and we named it RRD (RSSI, Rank and Dynamic). After taking into account hysteresis of the coverage zone of the transmission range of nodes, we optimized RRD. This enhanced version is called RRD+. Based on RRD+, we proposed MRRD+ (Multiple, RSSI, Rank and Dynamic) to support multiple sinks in convergecast LLNs in mobility. ADUP (Adaptive Downward/Upward Protocol) is a routing solution that supports both convergecast and anycast in LLNs concurrently.

We evaluated the performance of our contributions in both simulation using Cooja simulator and experiment (only for ADUP) on TelosB motes. The results obtained in both simulation and experiment confirm the efficiency of our routing protocols.

Keywords: wireless sensor networks, low-power and lossy networks, convergecast, anycast, mobility, multiple sinks.

Résumé

La technologie des réseaux de capteurs sans fil (RCSF) est l'un des éléments constitutifs de l'Internet des objets (IoT). En raison de leurs caractéristiques de déploiement facile et de leur flexibilité, ils sont utilisés dans de nombreux domaines d'application. Les réseaux à faible consommation et à perte (LLN) sont un type spécial de WSN dans lequel les nœuds sont largement limités en ressources. Convergecast est l'un des modes de communication de base, dans lequel tout le trafic du réseau est destiné à une destination prédéfinie appelée collecteur. Tout en prenant en compte les domaines d'applications IoT, convergecast n'est pas le seul mode de communication sur le réseau. Le récepteur doit envoyer des commandes à certains capteurs pour effectuer des actions. Dans cette application, anycast est un autre mode de communication de base. Dans anycast, le trafic provenant du récepteur est destiné à tout membre d'un groupe de récepteurs potentiels du réseau.

Les LLN sont formés de nœuds de capteurs statiques et changent rarement de position. En raison des contraintes de ressources strictes imposées au calcul, à l'énergie et à la mémoire des LLN, la plupart des protocoles de routage ne prennent en charge que les réseaux statiques. Cependant, la mobilité est devenue une exigence importante pour de nombreuses applications émergentes. Dans ces applications, certains nœuds sont libres de se déplacer et de s'organiser dans un réseau connecté. La topologie changerait continuellement en raison du mouvement des nœuds et de l'instabilité des liaisons radio. Il s'agit d'une tâche difficile pour la plupart des protocoles de routage des réseaux LLN afin de s'adapter rapidement au mouvement et de reconstruire la topologie en temps voulu.

Le but de cette thèse est de proposer un support de mobilité efficace pour les protocoles de routage dans les réseaux LLN. Nous nous concentrons sur convergecast et anycast, qui sont les modes de communication les plus utilisés dans les réseaux LLN, dans les scénarios de réseau mobile.

Nous proposons un mécanisme d'amélioration, nommé RL (RSSI and Level), pour prendre en charge les protocoles de routage dans les réseaux LLN convergecast en mobilité. Ce mécanisme aide le protocole de routage à prendre des décisions plus rapides pour la détection de la mobilité et la mise à jour des voisins du saut suivant, mais souffre d'une surcharge importante. Nous proposons une gestion dynamique des messages de contrôle pour améliorer les performances de RL et l'implémentons en plus du protocole de routage pour réseau à faible consommation (RPL) et nous l'avons nommé RRD (RSSI, Rank and Dynamic). Après une prise en compte de l'hystérésis de la zone de couverture de la plage de transmission des nœuds, nous avons optimisé RRD. Cette version améliorée s'appelle RRD +. Sur la base de RRD +, nous avons proposé MRRD + (Multiple, RSSI, Rank et Dynamic) pour prendre en charge plusieurs puits dans les réseaux LLN convergecast en mobilité. ADUP (Adaptive Downward / Upward Protocol) est une solution de routage prenant en charge simultanément convergecast et anycast dans les réseaux LLN.

Nous avons évalué les performances de nos contributions à la fois en simulation avec le simulateur Cooja et en expérience (uniquement pour ADUP) sur des motos TelosB. Les résultats obtenus en simulation et en expérience confirment l'efficacité de nos protocoles de routage.

Mot-clés: réseaux de capteurs sans fil, les réseaux à faible consommation et à perte, convergecast, anycast, mobilité, plusieurs puits.

Contents

List of Figures

List of Tables

List of Abbreviations

AODV	Ad Hoc On-demand Distance Vector Routing
AP	Access Point
BE	Backoff Exponent
BMP-RPL	Bayesian Model Mobility Prediction RPL
CCA	Clear Channel Assessment
C_ID	Corona ID
CPU	Central Processing Unit
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTP	Collection Tree Protocol
DAG	Directed Acyclic Graph
DAO	Destination Advertisement Object
DAO-ACK	Destination Advertisement Acknowledgement
dB	Decibel
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAGs	Destination Oriented DAGs
DSR	Dynamic Source Routing
ETX	Expected Transmission Count
FEDER	Financial Support of the European Regional Development Fund
GPS	Global Positioning System
IEEE	Institute of Electrical and Electronics Engineers
IETF ROLL	Internet Engineering Task Force Routing Over Low power and Lossy Networks
IoT	Internet of Thing
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
KP-RPL	Kalman Positioning RPL
LET	Logical Energy Tree
LLN	Low power and Lossy Network
LWB	Low-Power Wireless Bus
MAC	Medium Access Control
MANETs	Mobile Ad-hoc Networks
ME-RPL	Mobility Enhanced RPL
MF	Mobility Flag
mod-RPL	Modified RPL
MoMoRo	Mobility Support Layer
MP2P	Multipoint-to-Point
MRRD+	Multiple, RSSI, Rank and Dynamic
MT-RPL	Mobility-Triggered RPL
MTU	Maximum Transmission Unit
NB	Number of Backoff
ND	Neighborhood Discovery
OF	Objective Function
OSR	Opportunistic Source Routing

ORPL	Opportunistic RPL
P2MP	Point-to-Multipoint
P2P	Point-to-Point
RPL	Routing Protocol for Low power and Lossy networks
RRD	RSSI, Rank and Dynamic
RSSI	Received Signal Strength Indicator
SNR	Signal to Noise Ratio
TCP/IP	Transmission Control Protocol/Internet Protocol
TL	Time to Leave
μ**IP**	micro IP
WSN	Wireless Sensor Network
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks

Chapter 1

Introduction

1.1 Wireless sensor networks

Wireless Sensor Networks (WSNs) technology is attracting more attention with the emergence of the Internet of Things (IoT). Compared with wired networks, WSNs have the feature of easier deployment and better flexibility. With the rapid development of sensors, WSNs will become the key technology for IoT. A WSN consists a number of low-cost sensor nodes, which act as data generators or data relays. Each node is equipped with four main components: a sensor unit for data sensing, a microprocessor used to process data, a transceiver for data transmission, and finally a battery unit with limited capacity for power supply. These sensor nodes are capable of sensing the physical environment, collecting and processing sensed data, and communicating with each other in order to accomplish certain tasks. Due to the fact that sensor nodes have limited communication coverage, data needs to be sent in a hop by hop manner to reach the destination. This operation is managed by the routing protocol that is based on a certain routing metric.

WSNs have been widely developed in the last decade, and new applications are emerging rapidly. Most of these applications are used for surveillance and recording of environmental or physical conditions [1]. In these applications, multipoint-to-point (MP2P), usually called convergecast, is the basic traffic mode, where all traffic in the network is destined to a predefined destination called the sink node in a multi-hop manner. Figure 1.1 presents an example of convergecast WSN. The transmission direction of packets is upward from sensor nodes to the sink.

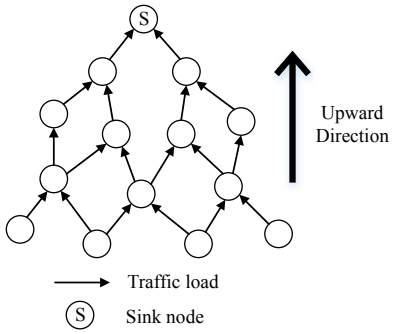
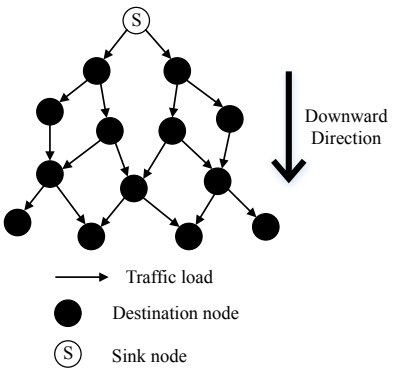
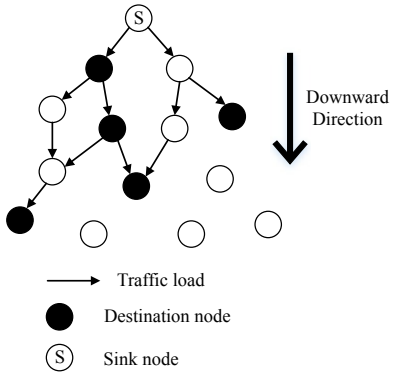


FIGURE 1.1: An example of convergecast WSN.

However, if we consider the Internet of Things (IoT) application domains, MP2P is not the only traffic mode in the network. The sink node needs to send commands to certain sensors or actuator nodes to perform actions according to received information. In these applications, point-to-multipoint (P2MP) and point-to-point (P2P) are two basic traffic modes. P2MP refers to traffic pattern that is accomplished by

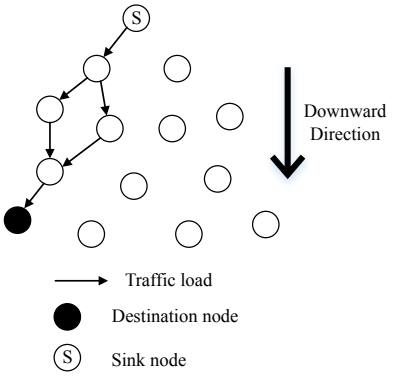


(A) An example of broadcast WSN.

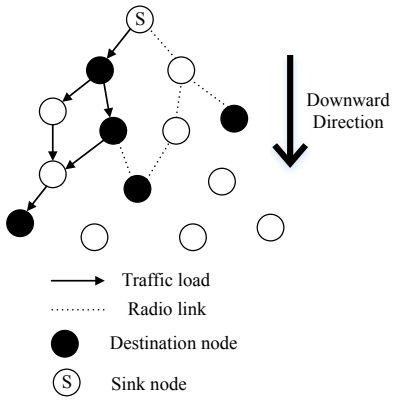


(B) An example of multicast WSN.

FIGURE 1.2: Examples of P2MP WSNs.



(A) An example of unicast WSN.



(B) An example of anycast WSN.

FIGURE 1.3: Examples of P2P WSNs.

broadcast or multicast, offering multiple paths from a single location to multiple locations. Figure 1.2 presents an comparison between broadcast WSN and multicast WSN. Unlike broadcast, multicast is group communication where data transmission is downward to a group of destination nodes.

P2P refers to a communication between two nodes; one node needs to transmit data to another. In this traffic mode, unicast and anycast are the two main network addressing methodologies. Unicast is communication between a single sender and a single receiver over a network, while anycast refers to communication between a single sender and any single member of a group of potential receivers in the network. Figure 1.3 presents a comparison between unicast and anycast communication. The transmission direction of packets is downward from the sink to a certain node.

1.2 Mobility in WSNs

Traditionally WSNs are formed by static sensor nodes that rarely change positions. However, with emergence of new applications, mobility has become an important requirement. In some scenarios, the monitoring nodes are mobile such as disaster response applications where a forest keeper should be able to request event related data from sensor nodes while moving inside a forest [2]. In other scenarios, objects being monitored need to be mobile, like animal monitoring applications. Sensor nodes attached to cows moving inside a field send information towards sink nodes

[2]. The degree of mobility varies in various applications. In some applications, only part of the network nodes need to be mobile, such as farming equipment monitoring [3]. In a farm, some equipment are installed in fixed points and some equipment are mobile like a plough. Whereas in some applications it is required that all nodes are mobile, such as health-care monitoring. Figure 1.4 shows an example of health-care monitoring system. In this example all patients are equipped with sensor nodes to transmit data to two deployed sinks and free to move inside a hospital [4]. The sinks are connected to the Internet and the doctor can get access to the patients data through a computer. In this example, there is more than one sink used to collect information. This is due to the fact that having one sink in the network will make paths from nodes to the sink longer as they move further away from the sink. Adding more sinks in the network would reduce the overall delay and path lengths to reach the destination.

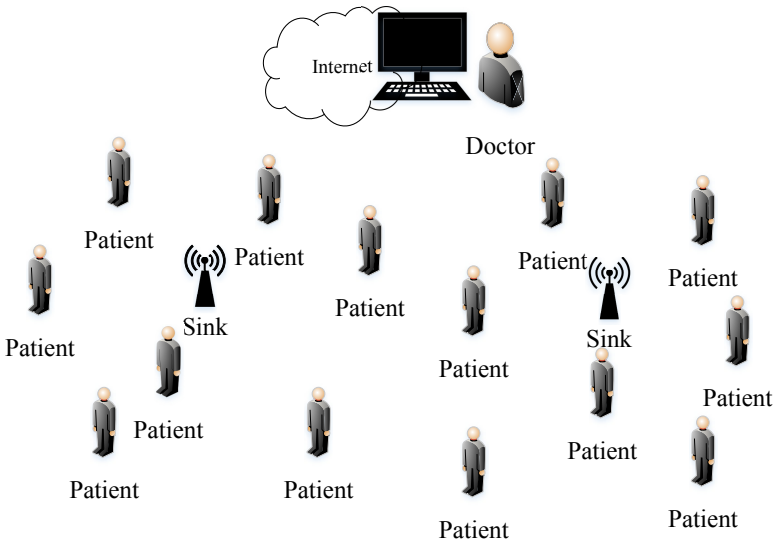


FIGURE 1.4: An example of health-care monitoring system.

In mobile scenarios nodes are free to move and organize themselves into a connected network. Hence, the topology is continuously changing due to the movement of nodes and radio links instability. This is a challenging factor for routing protocols. Every time the topology changes, the routing protocol needs to update the path to reach the destination. However, most routing protocols of WSN cannot be able to adapt rapidly to the topological changes and reconstruct routes in a timely manner. This will result in packet loss during transmission.

1.3 Routing protocols for low power and lossy networks

In recent years, Low-Power and Lossy Networks (LLNs) applications are gaining an interest from both research and industry. LLNs are a special type of WSNs in which nodes are largely resources constrained [5]. These nodes have limited processing power, memory and energy. They communicate over unstable links with low data rates and their state is usually unstable with low packet delivery rates. To cope with the resource limitation, the Institute of Electrical and Electronics Engineers (IEEE) created a new Medium Access Control (MAC) protocol, IEEE 802.15.4., with low MTU (Maximum transmission unit) size (127 bytes) and low data rate (250 kbps) [6].

A number of routing protocols have been proposed to provide the routing support for LLNs. Some of them are based on the ad hoc on-demand distance vector routing (AODV) [7]. These protocols are AODVjr [8], LoWPAN-AODV [9], NST-AODV [10], etc. These routing protocols make simplification on AODV to be suitable for the resource constraint and dynamic network environment. Collection Tree Protocol (CTP) [11], Hydro [12], and Routing Protocol for Low Power and Lossy Networks (RPL) [13] are also routing protocols for LLNs. Among these protocols, RPL is considered as the most promising routing protocol for LLNs because of its comprehensive features and great flexibility.

RPL is designed by Internet Engineering Task Force Routing Over Low power and Lossy networks (IETF ROLL) working group for LLNs. RPL adapts IPv6 (Internet Protocol version 6) and runs on top of IEEE 802.15.4 standard. RPL is designed to meet the requirements of many applications, which are mainly suitable for static networks. Supported traffic flows by RPL include MP2P, P2MP and P2P.

1.4 Motivation

Mobility support has become more and more important in various emerging IoT applications. The problem of mobility has been extensively studied in Mobile Ad-hoc Networks (MANETs). However, these solutions are often too complex to be adequate for LLNs. Since LLNs are strict resource restrained networks that consist largely of constrained nodes. The nodes have limited processing power, memory, and sometimes energy; and interconnection are lossy links, typically only supporting low data rates. Movement of nodes in LLNs would make the link more lossy and unstable that results in relatively low packet delivery ratio.

Although different mobility extensions for data convergecast in LLNs have been proposed, most of them can only support a few number of nodes in the network to be mobile. When only few nodes are mobile in a highly connected static network, there will not be a great impact on the topology. Indeed, when only few nodes change positions, most nodes do not need to rebuild the routes to reach the destination. It is a challenge to propose a mechanism that can support LLNs routing protocols in highly mobile convergecast scenarios in a confined space.

In contrast, the communication in the opposite direction also called anycast (sink to any leaf node) is more difficult to maintain. Indeed, as the mobile node moves from one position to another, downward routes established through control messages become out-of-date much more quickly. As a result, downward communications are highly unreliable due to inaccurate route information. Indeed, mobility support for anycast is also less studied than convergecast in LLNs.

In addition, more and more applications require multiple sinks rather than one sink. This is due to the fact that in a convergecast WSN only one sink easily leads to a faster energy depletion, more packet loss, higher latency and smaller network range. Deploying multiple sinks in the network can help solve these problems. However, there are very few research work on supporting both mobility and multiple sinks at the same time. Multi-sink support will be a critical topic in mobile LLNs.

The goal of this thesis is to propose an efficient mobility support for routing protocols in LLNs. We mainly focus on scenarios where the links are unstable and prone to failure. This is typically the case of mobile networks deployed in confined spaces. In these cases, a small change in the position of nodes might create a significant

change in the link quality. We also focus on convergecast and anycast mobile network scenarios, since they are the most used traffic modes in IoT. Besides, multi-sink support in mobility is another goal in this thesis.

1.5 Main contributions

This thesis focuses on the mobility support of routing protocols in Low power and Lossy networks (LLNs) and our main contributions are related to the network layer.

The key contributions are summarized as follows:

- Contribution 1: Mobility enhancement mechanism RL for convergecast WSNs.

  We propose an enhancement mechanism, called RL (RSSI and Level), to routing protocols to support mobility in convergecast WSNs. RL is based on two methods, Received Signal Strength Indicator (RSSI) monitoring and level (See section 3.1.1 in Chapter 3) updating. This mechanism helps routing protocol make faster decisions for detecting mobility and updating next-hop neighbors but suffer from high overhead.

- Contribution 2: Mobility enhancement RRD for RPL.

  RRD (RSSI, Rank and Dynamic) is an updated version for RL to be used on top of RPL. It is based on three methods: RSSI value monitoring, Rank (See figure 2.1 in Chapter 2) updating and dynamic control message management. Compared to RL, RRD not only enhances the network performance, but also reduces the network overhead.

- Contribution 3: Mobility enhancement RRD+ for RPL.

  RRD+ is an enhanced version for RRD that take into account hysteresis of the coverage zone of the transmission range of nodes. Indeed, when nodes are close to the edge of transmission range, they will suffer from frequent Rank updating, which may cause parent nodes loss. When applied to RPL and compared with other existing RPL mobility supports, results show that RRD+ enhances the ability of RPL to better cope with mobility scenarios.

- Contribution 4: Mobility enhancement MRRD+ for RPL with multiple sinks.

  MRRD+ (Multiple, RSSI, Rank and Dynamic), which is an enhancement over RRD+ takes into account multiple sinks. MRRD+ improves dynamic control message management in order to inform Rank updates as soon as possible in the case where a node is moving out of its parent node range.

- Contribution 5: Mobility enhancement mechanism ADUP for anycast WSNs.

  We proposed a routing protocol called ADUP (Adaptive Downward/Upward Protocol) that supports both convergecast and anycast in WSNs concurrently. Mobility support of convergecast is achieved according to RRD+, while mobility support of anycast is done by the sink node. ADUP can help the sink adapt to mobility quickly and build route paths to any node in the network in a timely manner.

1.6 Structure of the thesis

The manuscript is organized in five chapters. The first chapter presents an introduction to WSN, mobility in WSNs, routing protocols for LLNs, the motivations

and contributions of this thesis. The second chapter presents a mobility support in convergecast and anycast routing protocols, as well as muti-sink support in LLNs. The third chapter presents the contributions of this thesis. We begin with mobility enhancement mechanisms for data collection and data dissemination. Then we present RRD and its enhanced version RRD+. In succession multi-sink mobility enhancement MRRD+ will be presented. Convergecast and anycast mobility support ADUP will be presented at last. In Chapter 4, we present the performance evaluation in both simulation and experiment (only for ADUP) of our contributions where we compare obtained results with some existing protocols in the literature. Finally, Chapter 5 concludes this thesis by summarizing our contributions and opening up some perspectives to forward this work.

Chapter 2

Related works

This chapter focuses on different techniques that support mobility for the routing protocols in LLNs. These techniques can be classified into three groups, mobility support in convergecast routing protocols, mobility support in anycast routing protocols and multi-sink support in LLNs. Firstly we present the overview of RPL in section 2.1. Since most mobility support is proposed to be used on top RPL, it is necessary to realize the work process of RPL in advance. In section 2.2, we introduce mobility support in convergecast routing protocols. One recapitulative table is used to compare different techniques. Section 2.3 presents mobility support in anycast routing protocols and the other recapitulative table is used for comparison. In section 2.4, we introduce multi-sink support in LLNs. At last we conclude this chapter in the section 2.5.

2.1 RPL protocol overview

RPL is a distance vector routing protocol for networks with constraints on processing power, memory, and energy. During network construction phase, RPL builds a Directed Acyclic Graph (DAG) topology. All edges of the DAG are oriented in such a way that no cycles exist. This graph is partitioned into one or more Destination Oriented DAGs (DODAGs) and RPL calls these destination nodes as DODAG roots. To maintain the topology and exchange routing information, RPL defines four types of control messages: DIO, DAO, DAO-ACK, and DIS. A DIO (DODAG Information Object) message is used to carry information that allows a node to discover its neighbors. A DAO (Destination Advertisement Object) message is used to propagate destination information by a child node to a selected parent node. A DAO-ACK (Destination Advertisement Acknowledgement) message is sent in response to a DAO. A DIS (DODAG Information Solicitation) message is mainly used to probe its vicinity by soliciting a DIO message from a neighbor node. Within a DODAG, RPL uses an Objective Function (OF) to select and optimize routes according to different metrics.

The network is waken up by a certain node in the network. DIS is broadcast by this node to explicitly solicit the DIO messages from the neighbor nodes. Once the sink node begins to broadcast DIOs, the construction of the initial topology of the network starts. Each node that receives these messages builds a list of potential next-hops that we call parents set. The Rank of a node is a value that defines the position of the node with respect to the sink in terms of metrics in DIO messages. During the routing process, a Rank metric is used to avoid loops in addition to the routing metric. The Rank of the sink node is ROOT_RANK that contains a value of MinHopRankIncrease, which is the minimum increase of the Rank between a node and any of its parent nodes. The Rank value is proportional to the increase of the metric, therefore we get Rank computation of the node itself, which is shown in Equation 2.1.

$$Rank = ROOT\_RANK + a * MinHopRankIncrease \quad (2.1)$$

Where a is another metric value in DIO messages that comes from a node with a lower Rank.

By Equation 2.1 a node computes the Rank value and then broadcasts the Rank value using a DIO message. If the Rank of the receiver node is higher, it means this control message is from a potential next-hop. Note that the receiver node adds all senders with lower Ranks to its parents set. Figure 2.1 shows an example of RPL topology with different Rank values. Note that the nodes that are closer to the sink node have smaller Rank values and the sink node has the smallest Rank value.

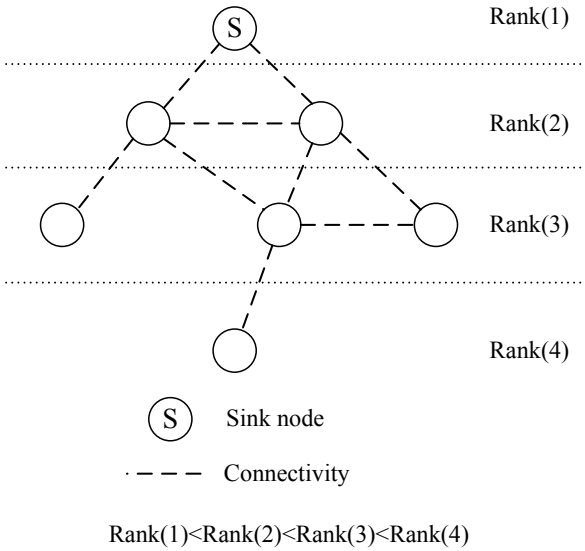


FIGURE 2.1: An example of RPL topology with Rank value.

RPL uses a trickle algorithm to disseminate DIOs over the network [14]. The aim of this algorithm is to reduce number of DIOs. Indeed, the rate of sending DIOs is reduced when there are no changes detected in the topology. Trickle algorithm contains three main parameters I_{min}, I_{max} and k. At the beginning, a variable I will be set to a value in the range of $[I_{min}, I_{max}]$, where I_{min} is the minimum duration and I_{max} is the maximum duration separating two consecutive DIOs. The first DIO interval I_{Dio} will be randomly taken from the range $[I/2, I]$. In order to avoid depleting the energy of nodes in a low-power network, the redundancy constant parameter k has been defined. Nodes can retransmit a packet to the same node a maximum of k times. Whenever nodes receive a transmission, a counter c is used to record the number of transmissions. If I_{Dio} expires and counter c is less than k, nodes will transmit DIOs. After this transmission, nodes double the interval for the next transmission until I reaches I_{max}. If the transmission is inconsistent or c is larger than k, nodes reset I_{Dio} to I_{min}, c to zero and start a new interval.

2.2 Mobility support in convergecast routing protocols

RPL is mainly designed to meet the requirements of static networks. In a mobile network, RPL adapts to mobility through the propagation of DIOs. In fact, to update the topology change, RPL requires waiting the reception of a new DIO message which is sent following a trickle algorithm. This algorithm is used in static networks in order to reduce overhead. Indeed, it reduces control traffic generation rate when

the topology is stable. However, in mobility scenarios, trickle algorithm cannot disseminate information in a timely manner and fails to cope with the changes of the topology. Moreover, RPL is reactive against mobility. Within two consecutive DIOs, RPL only uses OF to select and optimize next-hops according to link metrics; while the update of link metrics comes from packet loss. Besides, RPL specifications do not offer methods to update the Rank in a timely manner. This causes loops when a parent node becomes a descendant node.

In [15], authors propose a mobility support layer (MoMoRo) that can be applied to existing data collection protocols. MoMoRo helps nodes gather neighborhood information and pass this information through a fuzzy logic estimator. MoMoRo examines link quality by a fuzzy-logic-based estimator. Furthermore, MoMoRo includes an active destination search scheme that allows disconnected mobile nodes to quickly reconnect. After a packet transmission failure to mobile nodes, MoMoRo transmits one more packet to the destination. If it fails, MoMoRo starts building a new route to the mobile node by broadcasting beacons and collecting replies from neighbors.

The authors implemented MoMoRo in RPL and evaluated the performance in an indoor testbed environment consisting of 30 Tmote Sky devices [16]. The results show that MoMoRo can help improve the performance of packet delivery ratio but suffers from a high number of overhead packets compared with standard RPL. Moreover, in MoMoRo mobility detection only depends on the packet transmission failure. This makes the network very passive to react to topology change.

In [17], authors proposed a handoff mechanism called smart-HOP, a handoff mechanism tailored for mobile WSN applications. In smart-HOP mechanism, mobile nodes monitor the link quality by receiving reply packets from the serving Access Point (AP) during data transmission phase. The reply packets contain the average RSSI, or signal to noise ratio (SNR), of the n packets received by the AP. A mobile node starts the discovery phase when the link quality goes below a certain threshold until it finds out an AP that is above a reliable threshold. Furthermore, they use 3 beacons to validate the stability of the AP that is selected. Smart-HOP relies on the availability of data traffic to switch APs which means that it is very dependent on the application scenario for timely switching. Although smart-HOP is proposed for WSN, a large number of fixed APs are required, which is resource waste in many WSN applications.

In [18], authors integrate a proactive hand-off mechanism smart-HOP [17] in RPL and called it mRPL. mRPL aims to provide the mobile nodes with a lower hand-off delay and higher reliability. Additional timers are used to increase parent switching efficiency and reliability. Figure 2.2 shows the timing diagram of the smart-HOP mechanism. During data transmission phase, a mobile node is assumed to have a reliable link with an access point (AP). The mobile node monitors the link quality through the receiving packets from the serving AP. In the case that the average RSSI value is smaller than a threshold T or no packets are received, the mobile node will change to discovery phase. A burst of DIS message will be broadcast to solicit DIO messages from neighbor APs, and the mobile node would select the best AP and go to data transmission phase.

Simulation results using Cooja simulator show that mRPL outperforms standard RPL on packet deliver ratio and overhead. Although they succeeded in avoiding

centralized hand-off decision in APs, they still rely on a large number of fixed APs.

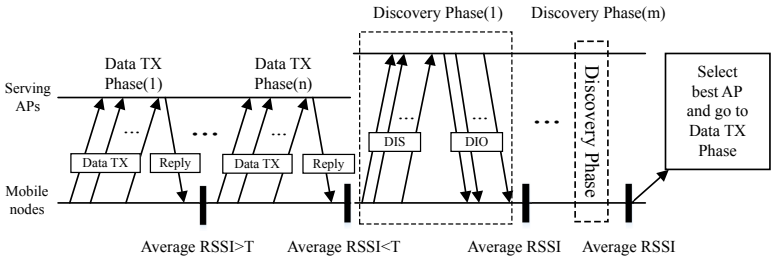


FIGURE 2.2: Timing diagram of the smart-HOP mechanism.

Hossein et al. [19] propose a mobility management framework named as mRPL+, which is an enhanced version of mRPL. Compared with mRPL with only hard hand-off mechanism, mRPL+ supports both soft and hard hand-off mechanism. In a hard hand-off process, a mobile node has to break a link before finding a new link, while in a soft hand-off process a mobile node select before disconnecting from the current one. mRPL+ runs soft hand-off as default and the hard hand-off is used once on response is received from the serving AP.

Performance evaluation using Cooja showed that mRPL+ is able to provide good reliability with lower hand-off delay compared with mRPL. However, the main drawback of mRPL+ is the same as mRPL that it relies on a large number of fixed APs to cope with mobility.

In [20], authors proposed a new cross-layer protocol called Mobility-Triggered RPL (MT-RPL). This protocol benefits from the X-Machiavel MAC protocol [21], which is an extension of the X-MAC protocol [22]. In MT-RPL a mobile node is known before and sets a flag for data packets that is used to prioritize transmissions from mobile nodes. According to X-Machiavel, on a busy channel mobile nodes can steal the medium of an ongoing transmission to send their packets first, and on an idle channel packets sent by mobile nodes can be opportunistically forwarded by static nodes.

Authors used WSNet [23] a discrete event simulator to evaluate their work. The results show that MT-RPL reduces the disconnection time and increases the packet delivery ratio. However, the main drawback is that MT-RPL increases the number of re-connections, and increases the number of DAO to report each parent change. This causes huge overheads compared with standard RPL, which is shown in results analysis in this paper.

Sneha et al. [24] proposed a mobility enabled RPL protocol to enhance hand-off process, optimize control messages and improve best parent selection scheme. In their context, they considered that mobile nodes and static nodes co-exist within the network. In order to make reliable transmissions, they use different timers to manage the Neighborhood Discovery (ND) process and apply average RSSI value to identify link quality. Within certain duration, if a robust link is found by a mobile node, best transmission route will be constructed, otherwise ND will be initiated. When mobile nodes detect no packet received within certain duration, DIS will be broadcast to call responses of neighbor nodes. Based on the average RSSI value of received DIO, mobile nodes detect a best route, otherwise ND will be initiated. In order to avoid loops, they set the Rank of mobile nodes as infinity Rank in order to solve the Rank problem when nodes move far away from the sink.

Performance evaluation using Cooja showed that the proposed mobility enhances RPL on packet delivery ratio and end-to-end delay. However, infinity Rank means this mobile cannot be selected by other nodes as a next-hop, which will packet loss and disconnected nodes when there are many mobile nodes in the network.

Gara et al. [25] proposed an adaptive timer algorithm in RPL called modified RPL (mod-RPL) to enhance the trickle timer to fit with mobility requirements. It takes into consideration the random trajectory of mobile nodes, pause time and the velocity of nodes. Each mobile node knows its own position and based on the position information the minimum Time to Leave (TL), which represents the minimum time to move out the radio range of its preferred parent, can be computed. Based on TL, if an inconsistency is detected, mobile nodes will reset the trickle timer. If a mobile node is suspected to be disconnected its parent nodes, an immediate DIS will be broadcast to solicit DIO from neighbor nodes.

The results evaluated by Cooja showed that mod-RPL outperforms RPL in certain scenarios. However, position information is only accessible without Global Positioning System (GPS) system which consumes much energy in LLNs. Mobility support with GPS is out of our research. Moreover, if the preferred parent of the mobile node is mobile as well, this algorithm cannot calculate an accurate TL due to unknown movement direction of the mobile preferred parent.

Somaa et al. [26] proposed Bayesian model Mobility Prediction RPL (BMP-RPL) aiming to adapt native RPL to mobile scenarios. They used a Bayesian model for mobility prediction, which is proposed in [27], to estimate the nodes connecting period according to a velocity probability distributions. This model estimates link duration between two nodes and introduces it as a new link quality metric. They proposed a new control message Hello protocol to broadcast link information. However, the degree of accuracy of prediction depends on the velocity distribution. Being able to predict the movement of all nodes in the network is something very rare and only few applications would be able to provide this information. In addition, BMP does not consider signal attenuation, which is another influence factor on connecting duration, in its model. Furthermore, it is challenging to implement Bayesian model for mobility prediction in LLNs considering sensor node constraints in terms of central processing unit (CPU) and energy.

Authors in [28] introduced a routing strategy called Kalman Positioning RPL (KP-RPL) for WSNs with both static and mobile nodes. This mechanism is based on location estimation using RSSI measurements. This approach improves the reliability and robustness of the network according to simulation results.

The performance of KP-RPL evaluated using Matlab [29] showed that the reliability and the robustness of the network in environments with harsh channel conditions are enhanced compared with geographical routing. However, the accuracy of KP-RPL relies on the path loss model, which is difficult to estimate in changing and real life deployments.

Cobarzan et al. [30] proposed a new version of the trickle algorithm (reverse trickle timer) in order to allow nodes to move seamlessly into a routing topology and limit overhead at the same time. They modified DAO by introducing a Mobility Flag (MF) in order to help other nodes identify mobile nodes. Compared with the standard trickle algorithm, reverse trickle timer starts from the maximum interval between two consecutive DIOs. They argue that the more the mobile nodes spend

time connected to the same parent, the more it is likely to move outside the coverage of the parent. Therefore, after each DIO, reverse trickle timer is divided by two until the minimum interval is reached. However, reverse trickle timer only makes sense when the mobile connects to a new parent and remains connected to this parent for a long time. When nodes move away from this parent, reverse trickle timer will cause a delay on updating information. In order to avoid congestion, standard RPL sets a delay for DAO before sending it. However, the authors did not take this delay into account. This delay will cause DAO messages loss when a node moves out the range of its parent node during this delay period.

El Korbi et al. [31] proposed a mobility support called Mobility Enhanced RPL (ME-RPL). ME-RPL is mainly based on identification of mobile nodes and dynamic control message management. They modified DIO messages by adding mobile identifier in order to help nodes identify mobile nodes. A node is more likely to choose a fixed node as its preferred parent by checking the identifier in DIO even if the Rank of mobile nodes is lower than the static nodes. Moreover a dynamic DIS management is proposed in order to update topology information according to nodes movement. Authors argue that if the preferred parent changes frequently, the node is in an unstable environment and the next DIS period should be shorter than the previous period in order to quickly obtain topology information. If the preferred parent stays the same, the next DIS period should increase. However, frequent changing of parent node will broadcast many DIS messages in a short period and causes a large number of DIO messages, which results in large overhead and network congestion.

In table 2.1 we used an array to summarize the mobility supports in convergecast routing protocols, where each protocol is presented by its acronym or its reference number. We compared the following 6 characteristics of all presented protocols:

- The method of mobility detection (Mobility detection) helps to know that by which way the protocol detects the movement.
- The method coping with mobility (Mobility solution) helps to know the solution used by the protocol coping with mobility.
- The mobility degree (Mobi. degr.) of scenarios shows the percentage of mobile nodes in the network that protocols cope with. High degree of mobility (High) means that most nodes in the network are mobile. Medium degree of mobility (Med.) means that more than one third nodes in the network are mobile. With low degree of mobility (Low) only a few nodes are mobile in the network.
- The number of control messages (Overh.) shows the cost of overhead and it can be classified in three grades: high (High), medium (Med.) and low (Low).
- The responsiveness to mobility (Resp.) helps to show the efficiency of routing protocol in mobile scenarios. If the protocol can detect mobility in advance and proactively helps nodes cope with mobility, its responsiveness to the mobility is high (High). In the case that the protocols is delayed to detect and cope with mobility, its responsiveness to mobility is medium (Med.). If mobility detection is delayed to the movement and mobility solution comes after a number of packets loss, the responsiveness to mobility is low (Low).

- The type of routing protocols (Type) for mobility can be classified in reactive (Reac.) and proactive (Proac.). Reactive means that a routing protocol is reactive to prevent packet loss in mobility, while proactive means that a routing protocol is proactive to detect mobility and avoid packet loss in advance.

Observing table 2.1, we notice that except standard RPL, MoMoRo and [30], all other protocols are proactive to detect mobility before packet loss. Indeed, detecting mobility in advance can avoid the large number of packet loss and reduce the number of retransmissions. Most proactive protocols detect mobility through the propagation of control messages. These protocols modified trickle algorithm in order to response the topology change quickly. However, the protocols with high responsiveness to mobility requires frequent control messages, which would cause high overhead in the network.

The comparative study reveals that none of these protocols can support highly mobile scenarios; only few nodes are free to move in their scenarios. When only few nodes are mobile in a highly connected network, there will not be a great impact on the topology. Indeed, when only few nodes change positions, most nodes do not need to rebuild the routes to reach the destination. In the case of high degree of mobility, the changes in the topology have greater impact on routes.

In our contribution, we concentrate on highly mobile scenarios where a significant part of the network or all nodes are free to move. The sink node that is fixed and does not change its position. Our work is based on signal strength monitoring and depth updating. It helps routing protocols to cope better with topology changes and makes proactive decisions on updating next-hop neighbors. Beside, a dynamic management of control messages is proposed in order to reduce the overhead in the network.

2.3 Mobility support in anycast routing protocols

Downward routing is less studied than upward routing in WSNs. It can be mainly classified into three categories: unicast based, broadcast based, and broadcast and unicast based. Most of these protocols are proposed for static scenarios and only few of them can cope with mobile scenarios.

RPL is a unicast based routing protocol that supports upward and downward traffic patterns. RPL supports two modes of downward traffic: Non-Storing mode and Storing mode. DAO messages are used to propagate destination information by a child node to the sink or a selected parent node. In Non-Storing mode, DAO messages are directly sent to the DODAG root along a default route as shown in figure 2.3. The root should establish source routing table entries for destinations learned from DAOs. Before sending a data packet, the root will use source routing to completely specify the route of the packet. In Storing mode, DAO message is unicast by the child node to the selected parents except the sink node to inform them of the existence of a child node as shown in figure 2.4. Nodes received DAO store downward routing tables for their sub-DODAG. All non-root and non-leaf nodes must store routing table entries for destinations learned from DAOs. Once a data packet is sent by the DODAG root, it must be sent to all one-hop neighbors first. Afterwards, the data packet will be sent hop by hop until it reaches its destination or hop limit. However, it is hard for RPL to support downward routing well in mobility. Due to topology changes in mobility, parent nodes will change frequently. This requires RPL to send DAOs timely in order to keep the downward routing table up-to-date, which will cost too much overhead. Otherwise, packets will be lost due to

TABLE 2.1: Summary of proposed mobility supports for convergecast routing protocols.

Protocols	**Mobility detection**	**Mobility Solution**	**Mobi. degr.**	**Overh.**	**Resp.**	**Type**
RPL[13]	Packet loss	Trickle algorithm	Low	Low	Low	Reac.
MoMoRo[15]	Packet loss	Link estimation	Med.	High	Med.	Reac.
mRPL[18]	Control messages	Immediate DISs and DIOs with hard hand-off	Low	Med.	High	Proac.
mRPL+[19]	Control messages	Immediate DISs and DIOs with hard and soft hand-off	Low	Med.	High	Proac.
MT-RPL[20]	Mobile flags in data packets	Taking possession of a reserved medium	Med.	High	High	Proac.
[24]	Control messages	Link estimation	Med.	Med.	High	Proac.
mod-RPL[25]	TL adaptive timer	Immediate DISs	Med.	High	Med.	Proac.
BMP-RPL[26]	Mobility prediction using Bayesian mode	Immediate Hello messages	Med.	Low	Med.	Proac.
KP-RPL[28]	Localization prediction	Kalman positioning	Med.	Med.	Med.	Proac.
[30]	Control messages	Reverse trickle algorithm	Low	Low	Low	Reac.
ME-RPL[31]	Control messages	Dynamic DIS management	Med.	High	High	Proac.

outdated path information.

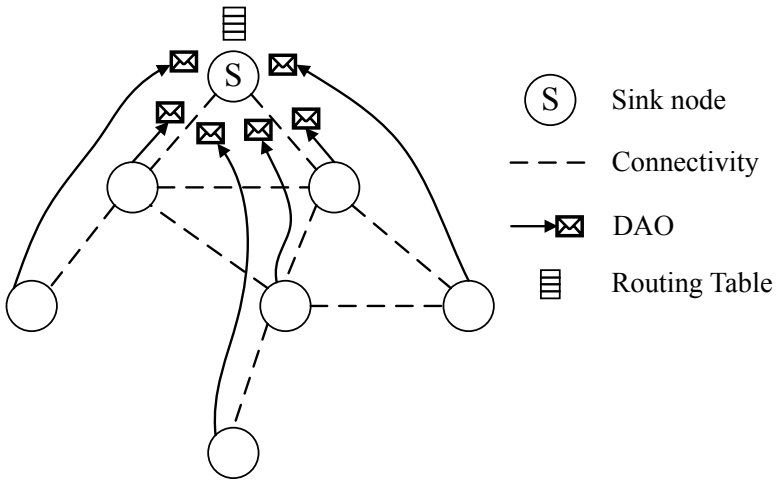


FIGURE 2.3: RPL Non-Storing mode.

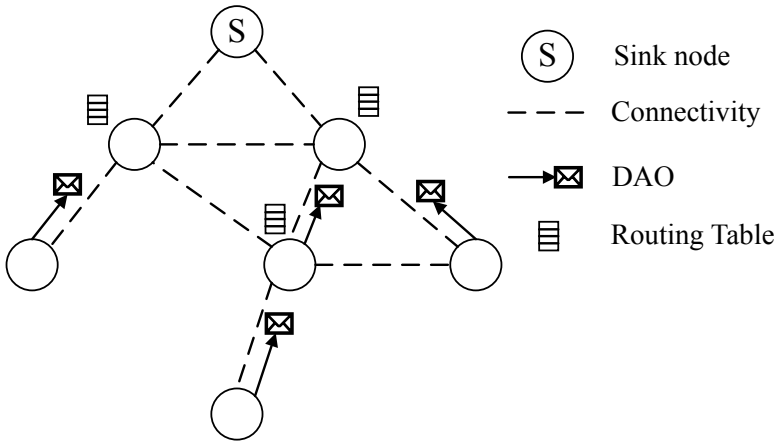


FIGURE 2.4: RPL Storing mode.

Carels et al. [32] proposed a new mechanism to improve RPL downward route updating of storing mode in mobility. In RPL a child node would unicast a No-path DAO to its parent node once the path lifetime expires. However, a No-path DAO would be lost when the mobile node is beyond reachable. Instead of sending a No-path DAO to a mobile parent node, authors propose that a child node sends a DAO containing No-path information to a static parent node to reach the sink first. The mobile parent node will be noticed the unreachable of connectivity with its child node from the sink later. By this way, the No-path information can be received by the mobile node successfully.

The work is implemented and evaluated using both simulation and experiment. The results show that it improves the end-to-end packet delivery ratio to mobile nodes from 20–30 % up to 80 % in grid topology with one mobile node. However, the main drawback is the mobility detection is based on packets loss and path lifetime, which is delayed and easily results in huge packets loss before finding new path. Moreover, it relies on the presence of fixed nodes. This means that this method cannot operate in a scenario where all nodes are mobile.

Due to inefficiency of DAOs in downward routing, Duquennoy et al. [33] proposed an opportunistic routing protocol called Opportunistic RPL (ORPL) which is built on-top of RPL. ORPL deactivates DAO and simply uses broadcasting of DIO. Each node owns a routing set that is a set of nodes with lower Ranks. This set will be propagated inside DIO messages. Routing set allows nodes to know whether a node is on a path to the destination or not. ORPL uses anycast instead of unicast to propagate a data packet. Nodes that receive the packet decide whether to forward it or not. If the destination of a packet is in the routing set, this packet will continue to be relayed, otherwise the packet will be dropped. ORPL is proposed for static scenarios only. When considering mobility, Rank and routing sets need to be updated timely according to the movement of nodes. Moreover, due to the fact that the updating of routing sets depends on the propagation of DIOs, timely updating will cost too much overhead and will not be practical.

DSR (Dynamic Source Routing) [34] and AODV (Ad hoc On-Demand Distance Vector) [7] are unicast based downward routing protocols. They both employ flooding methods to support route discovery and route maintenance. Compared with DSR, AODV further supports periodic advertisements and distance vector routing, which is more adaptive to dynamic scenarios. However, both DSR and AODV, need to run route discovery and route maintenance very often in order to update routing tables in a timely manner in mobile scenarios. In this process flooding of requests will cause congestion in the network and route discovery and maintenance cannot run well.

Improving AODV based on restricted broadcasting is a common method proposed in [35] for mobile scenarios. However, this method use geographic positions to assist the broadcasting to reduce the number of retransmissions, and thus require each node to be equipped with a Global Positioning System transceiver which is difficult to achieve in LLNs.

Ferrari et al. [36] proposed a flooding and time synchronized protocol for WSNs called Glossy. Glossy exploits the flooding mechanism to implicitly synchronize the network. According to clock values embedded in flooding packets, all receivers synchronize relatively to the clock of the source nodes. Each packet also embeds a relay counter value, which represents how many times a packet has been relayed. Nodes always transmit packets with the same relay counter concurrently.

Figure 2.5 illustrates an example of Gloosy transmission process. Benefiting from concurrent transmission and synchronization, Gloosy could avoid constructive interference, in which two waves (of the same wavelength) interact in such a way that they are aligned, leading to a new wave that is bigger than the original wave. However, Gloosy highly relies on concurrent transmissions and actuation of source nodes, and this results in Gloosy not supporting data collection scenarios. Since too many source nodes will cause too much concurrent transmissions simultaneously, which would cause serious congestion in the network; and congestion will induce inaccurate synchronization. Besides, in order to get precise synchronization, Gloosy has very strict packet size limit during transmission and this makes the propagation of variable-sized packets within a strict time control.

LWB (Low-Power Wireless Bus) [37] is an updated version of Gloosy, which supports one-to-many, many-to-one and many-to-many traffic concurrently. Unlike Gloosy that is simply driven by the events of source nodes, LWB appoints a node in

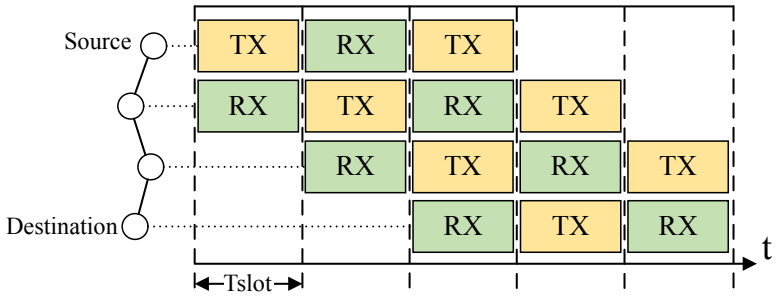


FIGURE 2.5: An example of Gloosy transmission process.

the network to work as a controller and it will send schedules of each source node. Although LWB reduces congestion using a central control method, it still meets the same problems of limited number of source nodes and packet sizes as Gloosy. In addition, both Glossy and LWB do not support mobility.

OSR (Opportunistic source routing) [38] is a broadcast and unicast based downward routing protocol which introduces opportunistic routing into traditional source routing. OSR uses a bloom filter mechanism a space-efficient probabilistic data structure to encode downward source route, which enables a reduction of the length of packet header while processing source routing. OSR uses multiple traffic flow patterns unicast, multicast, and broadcast. Unicast and multicast are the main ways to propagate information. In the case of failure of unicast and multicast, broadcast will be used. OSR achieves a reduction in transmission count and a gain in reliability compared to standard RPL.

Authors evaluated the performance of OSR via both simulations and real-world testbed experiments, in comparison with the standard RPL (both storing mode and non-storing mode). The results show that OSR significantly outperforms RPL and in scalability and reliability. However, OSR also needs route discovery and maintenance to deal with mobile scenarios, which will meet congestion problem similarly to DSR and AODV.

In table 2.2 we summarize the mobility supports in anycast routing protocols, where each protocol is presented by its acronym or its reference number. We compared the following 6 characteristics of all presented protocols:

- The type of routing protocols (Type) for mobility can be classified in Unicast, Broadcast and Broadcast and Unicast.
- The methods used to build downward route (Downward route building) show the way of protocols to build route path in mobility.
- The methods used to maintain downward route (Downward route reparation) help to know if the protocols can repair the downward route timely according to the topology change.
- The mobility degree (Mobi. degr.) of scenarios shows the percentage of mobile nodes in the network that protocols cope with. High degree of mobility (High) means that most nodes in the network are mobile. Medium degree of mobility (Med.) means that more than one third nodes in the network are mobile. Low degree of mobility (Low) signifies that only a few nodes are mobile in the network.

- The number of control messages (Overh.) shows the cost to build downward route and it can be classified in three grades: high (High), medium (Med.) and low (Low).
- The responsiveness to mobility (Resp.) helps to show the efficiency of routing protocol in mobile scenarios. If the sink proactively copes with mobility and builds the downward route in advance, the responsiveness to the mobility is high (High). In the case that the sink is delayed to cope with mobility, the responsiveness to mobility is medium (Med.). If the downward route building is delayed to the movement and numbers of packets are lost before this process, the responsiveness to mobility is low (Low).

Observing table 2.2, we notice that it is hard for a unicast based routing protocol to achieve high responsiveness to mobility. This is mainly due to the fact that unicast based routing protocols take much time on updating topology and building route path. Some protocols like [32] or ORPL modified DAO messages to reduce time cost on updating topology. The comparative study reveals that route discovery and route maintenance would cost much overhead with slow responsiveness to mobility. This is mainly due to the fact that source routing would propagate control messages to the entire network which costs a lot of resources. Furthermore, in mobility the topology changes fast, source routing needs frequent route discovery and maintenance to maintain the topology, which is hysteresis to build route path in mobile scenarios. Although [35] achieves high responsiveness to mobility, this is mainly because this method use geographic positions to assist updating topology; geographic information could help node quick detect movement and proactively build a new path to the destination.

We also notice that broadcast based routing protocols or broadcast and unicast based routing protocol achieve high responsiveness to mobility. The reason is that broadcasting is insensitive to mobility and routing protocols do not need to update topology according to movement. The flooding method broadcasts the packets to the entire network, which is easily to reach the destination. However, since flooding method takes much network resource, it is hard for a broadcast based method to support concurrent transmissions or large number nodes.

In our contribution, we propose an adaptive routing protocol that integrates point-to-point traffic and data collection traffic together and supports highly mobile scenarios. This protocol reacts quickly to the movement of nodes to make faster decisions for the next-hop selection in data collection and dynamically build routes for point-to-point traffic.

2.4 Multi-sink support in LLNs

Mobility support has become an important requirement in LLNs. Multi-sink support is already a well-known topic, but most studies are based on the MAC layer of IEEE 802.11, such as in [39] and [40]. In these studies position information of nodes is offered by GPS system, which is out research in this thesis. In addition, multiple mobile sinks are also used as data mobile data collectors to balance traffic load in order to prolong the network lifetime, like [41] and [42]. However, in this paper we only consider static sink nodes. There are very few research work on supporting both mobility and multi-sinks at the same time. According to our knowledge, only

TABLE 2.2: Summary of proposed mobility supports in anycast routing protocols.

Protocols	**Type**	**Downward route building**	**Downward route reparation**	**Mobi. degr.**	**Overh.**	**Resp.**
RPL[13]	Unicast	DAO messages	No-path DAOs and route lifetime	Low	Low	Low
[32]	Unicast	DAO messages	Modified No-path DAOs and route lifetime	Low	Low	Med.
ORPL[33]	Unicast	Modified DIOs and deactivated DAOs	Route lifetime	Low	High	Med.
DSR[34]	Unicast	Route discovery	Route maintenance	Low	High	Low
AODV[7]	Unicast	Route discovery	Route maintenance	Low	High	Med.
[35]	Unicast	Route discovery with restricted broadcasting	Route maintenance with restricted broadcasting	Low	Med.	High
Gloosy[36]	Broadcast	Flooding mechanism	Concurrent transmission	Med.	Low	High
LWB[37]	Broadcast	Flooding mechanism	Concurrent transmission with a central controller	Med.	Low	High
OSR[38]	Broadcast and unicast	Modified route discovery	Modified route maintenance	Med.	Med.	High

[43] has dealt with both issues simultaneously. Therefore, we only introduce multi-sink support mainly in static LLNs in this section.

Current support for multiple sinks of RPL is limited to a short description in the specification of RFC 6550 [13]. RPL supports two types of multi-sink scenarios as shown below.

- Multiple sinks with different DODAGIDs (Destination Oriented DAG IDs).
  - Different DODAGs with no connectivity to coordinate with each other (Nodes belonging to one DODAG cannot join another DODAG and they cannot coordinate with the nodes belonging to another DODAG.).
  - Multiple DODAGs acting as a means to partition the network (Nodes belonging to one DODAG cannot coordinate with the nodes belonging to another DODAG, but they are possible to join another DODAG.).
- Multiple sinks sharing the same DODAGID.

The first type is mainly considered for networks where there is no need to exchange data between different DODAGs. In the second type, the whole network shares only one DODAG, and all the nodes in the topology have the possibility to connect with each other. A virtual root is supposed to be connected to these multiple sinks over a reliable transit link. In this paper we only consider the second type.

In [44], authors proposed a method to support usage of multiple sinks for RPL. Multiple sinks share the same DODAGID and all of them are coordinated by a virtual root. The results tested by Cooja show that the maximal energy usage of the nodes decreased with about 45 % and the average energy consumption decrease with more than 30 %. In this paper, authors considered sinks with the same DODAGID as one sink. Therefore, when supporting many-to-one scenarios in RPL, their method did not support sink selection. However, even with the same DODAGID, a better sink selection method can help manage the residual energy of sinks and reduce collisions with a load balancing technique.

In [45], authors proposed a method for RPL to support multi-sink scenarios. This method uses the available bandwidth, delay, MAC layer queue occupancy and expected transmission count (ETX) as a union metric in conjunction with the shortest hop-count metric. In the network there are multiple DODAGs and each DODAG is identified with a different DODAGID. When a node receives a DIO message, it will use hop-count metric (Rank) to choose a sink first. If this DIO is from a lower Rank node, this node will be treated as a potential next-hop. In the case that DIO receiver has multiple parent nodes with the same Rank, tie-breaking metrics will be used to determine the best next-hop. The node will then join the DODAG of the chosen parent node.

The simulation results performed using Cooja demonstrate that their work increased the packet delivery ratio by up to 25 % and decreased the number of retransmissions by up to 65 % compared standard RPL protocol that only uses the hop-count metric. However, simple comparison of tie-breaking metrics will cause frequent changing of parent nodes. This changing may also cause the changing of DODAG and packets are likely to be lost during handover of different DODAGs.

In [46], authors proposed two routing algorithms that are based on a Logical Energy Tree (LET): one with centralized sink node called LETCSN and the other with centralized sink node and secondary sink nodes called LETSSN. The basic idea of LETCSN is that residual energy of sensor nodes is used as an additional parameter for making routing decisions. According to residual energy and distance to the sink, all nodes are divided into three levels. The sink node is set on Level 0. A threshold value of energy is used to fix sensor nodes at different levels. Nodes with energy more than the threshold value are set on Level 1 and nodes with energy less than the threshold value will be set on Level 2. Level 1 nodes have bidirectional links with sink node and Level 2 nodes, while Level 2 nodes have only unidirectional links with Level 1 nodes. Authors argue that this way, energy will be saved. However, it is likely for the sink node to be a bottleneck if the bandwidth of data increases. Therefore, they proposed using multiple sinks called secondary sink nodes, placed at fixed locations. However, these two algorithms will increase the average number of hops of a packet. Nodes that are close to the sinks will consume more energy than other nodes. After a period, these close nodes will be changed from Level 1 to Level 2 due to energy consumption. Because of unidirectional links constraints, these nodes need more hops to reach the sinks. Even with multiple sinks, this problem cannot be solved when the size of the network is large.

In [47], a genetic algorithm is proposed to solve the problem of load balancing amongst sinks in a multi-sink WSN. This algorithm works in two steps. In the first step, a chromosome, which is an array of sink selection for each node, will be generated. A fitness function by analyzing the load on each sink will be used to analyze the quality of each chromosome. In the second step, half of the population with the best chromosomes will be selected to construct parent pairs (A parent pair is a pair of chromosomes.). These parent pairs will apply crossover or mutation and get the feasible sink node in the end. This approach takes a lot of time and calculation to converge. In the first step, in order to get the sink selection information the node that is far away from the sink must recursively repeat the binding procedure until it gets this information. In the second step, if the size of population is big, it will take many iterations to get the best solution. This method may balance the energy consumption of each sink, but will increase energy consumption due to control traffic and delay in convergence.

Co-RPL is an extension to RPL based on the Corona mechanism to support mobility [43]. Co-RPL allows the DODAG root to generate DIO messages periodically instead of using a trickle timer. Authors modified DIO messages by adding a corona ID (C_ID), which is used as a relative coordinate to localize mobile nodes according to the DAG root. Nodes select a parent that has the smallest C_ID and then increment C_ID by one before broadcasting this value. Corona mechanism is similar to Rank mechanism using the hop-count metric, and C_ID is just like the Rank value in Rank mechanism. However, the same with Rank value, the updating of C_ID is not proactive. In Co-RPL DIOs are generated periodically. Therefore, C_ID updating frequency depends on DIOs interval. Thus, if nodes broadcast DIOs frequently, the network will suffer from high overhead. If there is no suitable connectivity between different DODAGs, packets can never reach to the next-hop nodes in case the next-hop nodes change their DODAG due to movement.

Aimed at maximizing the network lifetime, authors [48] proposed an energy optimized routing algorithm for multi-sink wireless sensor networks based on a combination of the number of hops and the residual energy of nodes. Avoid strategy for low-energy nodes and load balancing strategy for multiple sinks were applied to dynamically prolong the lifetime and achieve balanced energy consumption of nodes. If a source node has a data packet to transmit, it selects the sink to which it has the smallest link metric a hybid of number of hops and residual energy of nodes. Nodes less than 10 % of the initial energy should be reduced the opportunity to be selected as the relay node.

Simulation results show that the routing algorithm balances the energy consumption of nodes effectively and extends the network lifetime, compared with similar algorithms. However, the main drawback is that without considering congestion of network this algorithm is not optimal on end-to-end delay. Retransmissions result in a higher delay and extra energy consumption.

In a mobile single-sink scenario all nodes except the sink are free to move and transmit data to the sink in a continuous manner. Having one sink in the network will make paths from nodes to the sink longer as they move further away from the sink. Adding more sinks in the network would reduce the path lengths and reduce overall delay and traffic. However, most researches related to multi-sink concentrate on deployment strategy of sinks or optimizing sink hand-over process. There are very few research works on supporting multi-sinks in mobile scenarios. In our contribution, we propose an enhancement based on signal strength monitoring and Rank updating in order to improve the network performance in mobile multi-sink scenarios. This enhancement helps RPL to better cope with mobility scenarios and to make faster decisions on updating next-hop neighbors.

2.5 Summary

Dealing with mobility in LLNs is a challenging task for a compromise between efficiency and complexity. However, most routing protocols designed for LLNs were originally specified without any special support for mobility, like RPL. More and more researchers are attracted to design mobility supports for routing protocols to achieve specific applications.

Mobility support in convergecast scenarios is well researched and many works have been proposed to enhance the performance of RPL in mobility. However, none of these works supports highly mobile scenarios. Indeed, when only few nodes are free to move in the network, there is not a great impact on the topology. Moreover, the works with high responsiveness to mobility requires large number of control messages to update topology change, which cause high overhead in the network.

Mobility support in anycast scenarios is less studied. The same as mobility support in convergecast scenarios, none of these works supports highly mobile scenarios. Some works use route discovery and route maintenance to build route path. Source routing process is hysteresis to the movement of nodes and takes much time on updating topology. Some works using flooding method to achieve high responsiveness to mobility. However, it is hard to support concurrent transmissions. Since broadcasting takes much network resource.

Most multi-sink support methods for mobility use GPS system to get position information. GPS is out of research in this thesis. However, without geographical positions informations, these methods are hard to achieve in LLNs. According to

our knowledge, there are very few research works on supporting both mobility and multi-sinks at the same time.

Our main contributions in this thesis are proposing efficient mobility supports for routing protocols in LLNs. The supports can be used in highly mobile convergecast and anycast scenarios. Besides, multi-sink support in mobility is another contribution in this thesis.

Chapter 3

Contributions

In this chapter, we present our contributions. Our main contributions work in convergecast and anycast mobile scenarios. In section 3.1, we present RL mechanism which is a mobility enhancement is for convergecast WSNs. RL can be used in most routing protocols to enhance their performance in mobile scenarios. However, RL suffers from high overhead. In section 3.2 we describe RRD that runs on top of RPL. RRD is an enhanced version of RL that solves the high overhead problem of RL. After considering hysteresis of the coverage zone of the transmission range of node, we propose RRD+ and present it in section 3.3. In section 3.4, we present our contribution MRRD+. MRRD+ extends RRD+ to support multiple sinks. Section 3.5 presents ADUP, which is the mobility enhancement mechanism for convergecast and anycast WSNs. Mobility support of convergecast is achieved according to RRD+, while mobility support of anycast is done by the sink node. Finally section 3.6 summaries the chapter.

3.1 RL : mobility enhancement mechanism for convergecast WSNs

In many-to-one multi-hop wireless sensor networks, data is transmitted by sensor nodes towards the sink node which is the only destination. Before sending data, nodes need to select a next-hop based on one or more routing metrics. In mobile scenarios, changes in the topology of the network increase the risk of link failures. In addition, if the routing metrics cannot be updated timely, it is likely to cause loops. We propose an enhancement mechanism based on a combination of RSSI value monitoring and level updating that we called RL mechanism (RSSI, level). This mechanism makes faster decisions for updating next-hop neighbors. RL is based on level knowledge, link existence monitoring and movement direction estimation. When applied to traditional routing protocols, RL allows nodes to anticipate on link failures by monitoring the change in the RSSI values. RL copes with mobility faster, and thus, enhances the overall performance of the network.

3.1.1 Level mechanism and calculation

During the routing process, a level metric is used in addition to the routing metric. The construction of the initial topology of the network starts at the sink node that begins to broadcast control messages which contain routing metrics. Each node that receives these messages builds a list of potential next-hops that we call parents set in what follows. The level of a node is a value that defines the position of the node relative to the sink in terms of number of hops, with sink node being at level 0. Based on the hop count obtained from control messages, a node will know the level of sender

node. If the level of the receiver node is higher, it means this control message is from a potential next-hop. The level of the receiver node will be equal to the smallest received level plus 1. Note that the receiver node adds all senders with lower levels to its parents set. Our level mechanism is similar with the Rank mechanism of RPL using the hop count metric.

3.1.2 Metric diffusion

With RL mechanism, control messages have three functions: (i) broadcasting metrics, (ii) monitoring link existence, (iii) and monitoring movement direction. Control messages are sent periodically in order to update routing metrics timely. At the beginning of each period the sink is the first to broadcast a control message. When a node receives a control message from a node with lower level, it updates the routing metrics and then broadcasts its level and metrics. Nodes only broadcast control messages once they receive them from nodes with lower level. This insures broadcast process to have a downward direction from the sink to the leave nodes.

3.1.3 Link existence and movement direction monitoring

Every node in the network needs to make sure that the link between itself and nodes in the parents set exists. We use the RSSI values to allow nodes to monitor links [49]. Nodes obtain RSSI values from control messages. A node stores two RSSI values for each parent, Old RSSI value ($OldRSSI$) and New RSSI value ($NewRSSI$). We use them to monitor movement direction. $OldRSSI$ is obtained from the previous control message and $NewRSSI$ is obtained from the newest control message. We consider that the node is getting closer to its parent if $NewRSSI$ is bigger than $OldRSSI$. In case $NewRSSI$ is smaller than $OldRSSI$, we consider that the node is getting further away from its parent.

In addition, we set a RSSI threshold. We assume that within the threshold the node has a good link quality with its parent node. When the RSSI value is smaller than the threshold, the node will consider whether to stop using the link or not. If the node is getting near to its parent, we consider that this parent node can still be used as a next-hop; otherwise this parent will be deleted from the parents set.

Each time a node adds a new node into the parents set, a timer is set for this parent. We call this timer the lifetime of a parent. Nodes are kept in the parents set for the duration of their lifetime. When the lifetime of a parent expires, this parent node will be removed from the parents set. While before the lifetime of a parent expires, if a new control message is received from this parent, the lifetime will be reset and this parent will be keep in parents set. According to RL mechanism, we set two sorts of lifetimes: long lifetime and short lifetime.

Long lifetime ($Long\_Lifetime$) is given to nodes that have a RSSI value that is higher than the threshold, and short lifetime ($Short\_Lifetime$) is given to nodes that have a RSSI value that is lower than the threshold. Indeed, when the parent node is in a zone where the radio link is about to fail, a short lifetime will help avoid using this parent node for a long period in case we do not receive its control messages.

3.1.4 Loop avoidance and level update

A routing loop is a common problem in mobility scenarios. Due to the changes in the topology, a current next-hop may become a child node, and loops will occur. In order to avoid this, we set the following three rules. (i) The level of a node must

be greater than the level of all nodes in its potential parents set. (ii) Nodes cannot forward data packets to nodes with higher or equal levels. (iii) Nodes must ignore control messages which are received from higher level nodes.

Monitoring link existence and movement estimation allows nodes to update their levels. Algorithm 1 depicts the level update process.

$Level\_r$ stands for the level of the receiver of the control message and $Level\_s$ stands for the level of the sender of the control message. $THRESHOLD$ stands for the RSSI threshold.

```
begin
    if received a control message then
        if Level_r > Level_s then
            if NewRSSI ≤ THRESHOLD then
                if NewRSSI ≤ OldRSSI then
                    Level_r = Level_r + 1;
                else
                    Level_r = Level_s + 1;
                    Setting Short_Lifetime for this parent node;
                end
            else
                Level_r = Level_s + 1;
                Setting Long_Lifetime for this parent node;
            end
        else
            Ignore
        end
    end
end
```

Algorithm 1: Level Update.

Figure 3.1 shows a scenario with four nodes A, B, C and P. P is the neighbor of nodes A, B and C. D is the neighbor of node B. We use a dashed circle to represent the RSSI threshold of node P and a solid line to represent the transmission range. Note that due to the nature of wireless signal propagation, in reality both the RSSI threshold and the transmission range are most likely to look like a cloud that changes from one transmission to another. Indeed, in our simulation model we used a probabilistic propagation model to take into account links instability. A, B and C will execute algorithm 1 whenever they receive a control message from P.

If the level of receiver node $Level\_r$ is bigger than the level of sender nodes $Level\_s$, case of node A, B and C in figure 3.1, they need to check $NewRSSI$ of control message. If $NewRSSI$ is smaller than $THRESHOLD$, case of node B and C, they compare it to the $OldRSSI$. If $NewRSSI$ is bigger than the old one, case of node C, this means that node C is moving forwards node P. Since node C is not within the $THRESHOLD$, we only give $Short\_Lifetime$ for node P and updates its own level according to $Level\_r = Level\_s + 1$. Otherwise, if $NewRSSI$ is smaller than the old one, case of node B, this means that node B is getting away from node P. In this case, node B increments its own level by $Level\_r = Level\_r + 1$. In figure 3.1, the RSSI value of node A is bigger than the $THRESHOLD$, thus A sets $Long\_Lifetime$ for parent P and updates its level $Level\_r = Level\_s + 1$.

By this process, nodes can detect the movement of other nodes. We do this because based on RSSI values, a node does not know if it is the one that is moving

away or if it is the sender node that is moving away. In this example, if the sender node P is moving away, nodes A, B and C will eventually receive new control messages from other neighbors, which are not shown in the figure 3.1, and update their level according to these control messages. And in case it is the receiver node that is moving away (node B), incrementing its own level will help avoid that this node be selected as a parent by its children nodes case of node D. This way we avoid loops from occurring.

Finally, in case the *Level_r* is smaller than the *Level_s* we ignore the control message.

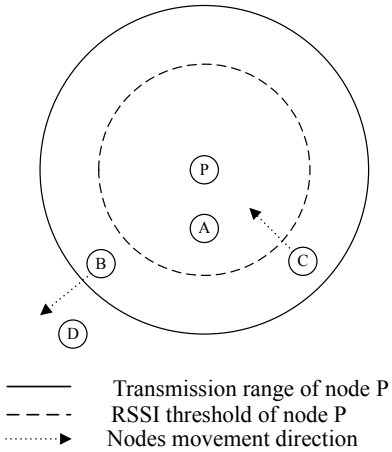


FIGURE 3.1: Possible situations between neighbor nodes.

3.2 RRD : Mobility enhancement for RPL

RPL is one of the most dominant routing protocols for LLNs. However, it is original specified without any support for mobility. Thus, we proposed RRD to support covergecast communication of RPL in mobility. RRD is an enhanced version for RL to be used on top of RPL.

If we only consider RPL in convergecast mobile scenarios, data is transmitted by sensor nodes towards one destination which is the sink node. According to RPL, before sending data, nodes need to select a next-hop based on a Rank value and routing metrics. However, standard RPL does not support timely Rank updates, a current next-hop may become a descendant due to untimely update of the Rank value, and loops will occur. Level mechanism of RL is similar with the Rank mechanism of RPL using the hop count metric. RL could help nodes to update level value in mobile scenarios. Thus, we propose to implement level update method in Rank mechanism of RPL and name it as RRD (RSSI, Rank and Dynamic). RRD can help RPL cope with topology changes when all nodes except the sink are mobile. RRD helps update Rank in a timely manner and allows nodes to have more than one potential next-hop option.

RPL uses DIO messages to diffuse metrics. The sending interval between continuous DIO messages is controlled by trickle algorithm. However, the trickle algorithm cannot cope with mobility timely. Besides, when using the trickle algorithm, DIO interval is common to all nodes and a node cannot adapt it to its own local topology changes. Periodical control packets broadcasting is a common method to help RPL update topology quickly. Indeed, we used periodical control packets broadcasting in RL. However, this method suffers from high overhead. Thus, in RRD

we propose a new DIO interval management algorithm, which copes with topology changes and reduces overhead. This algorithm dynamically modifies the frequency of DIOs according to Rank updates and allows each node to have its own frequency for DIO messages. Compared with RL, new method adapts to mobility faster with less overhead.

3.2.1 Common techniques between RL and RRD

In order to monitor link existence and update Rank, RRD follows approaches used in RL. In RL nodes obtain RSSI values only from control messages, whereas in RRD nodes obtain RSSI values from control message DIO and acknowledgement messages (ACK). At convergence stage, the network is constructed through DIO messages and RSSI values are obtained from these messages. After network construction, data packet senders will receive an ACK after each successful packet transmission. This way, senders will get continuous RSSI value updates through ACKs. When implement algorithm 1 in RPL, we get algorithm 2 shown as follow. $Rank_r$ stands for the Rank of the receiver of the control message and $Rank_s$ stands for the Rank of the sender of the control message. $Long\_Lifetime$ stands for the long lifetime and $Short\_Lifetime$ represents the short lifetime. $MinIncrease$ stands for the value of MinHopRankIncrease. Note that Rank update is similar with the level update. Indeed, level mechanism is the Rank mechanism of RPL using the hop-count metric.

```
begin
    if received a DIO message then
        if Rank_r > Rank_s then
            if NewRSSI ≤ THRESHOLD then
                if NewRSSI ≤ OldRSSI then
                    Rank_r = Rank_r + MinIncrease;
                else
                    Rank_r = Rank_s + MinIncrease;
                    Setting Short_Lifetime for this parent node;
                end
            else
                Rank_r = Rank_s + MinIncrease;
                Setting Long_Lifetime for this parent node;
            end
        else
            Ignore
        end
    end
end
```

Algorithm 2: Rank Update for nodes in mobility.

3.2.2 Ripple control message management

Trickle algorithm is used by RPL in order to reduce overhead in static networks. Trickle algorithm starts with a short interval between two DIOs in order to construct network fast. Each time the interval will be increased until it reaches I_{max}. However, when the interval increases, trickle algorithm cannot cope with information update

in a timely manner, which is important in mobility. In RL, we use periodical control message broadcasting to update topology. However, it costs too much overhead (See Subsection 4.2.6 in Chapter 4). Besides, when using the trickle algorithm or periodical control message broadcasting, DIO interval is common to all nodes and a node cannot adapt it to its own local topology changes. Therefore, we propose a new DIO interval management called ripple control message management, which copes with topology changes and reduces overhead. This algorithm dynamically modifies the frequency of DIOs according to Rank updates and allows each node to have its own frequency for DIO messages.

DIO messages are used to construct routes towards the sink. In other words, the main function of DIO messages is to help children nodes find parent nodes. A DIO message that comes from a smaller Rank is more important than a DIO message that is received from a higher Rank. This is mainly due to the fact that the diffusion of path metric is from lower Rank nodes to higher Rank nodes. This means that information contained in DIO messages coming from lower Rank nodes, will be the base for higher Rank nodes to update path information. Thus, the rational behind our algorithm is that if nodes with lower Rank values propagate DIOs timely, they will make sure path metrics used by higher Rank nodes are up-to-date. Therefore, nodes that are closer to the sink should send DIO messages more frequently and the frequency of DIO messages may be reduced for nodes with higher Ranks.

We designed a dynamic DIO interval calculation according to Rank updates. The DIO interval calculation is shown in equation 3.1.

$$DIO\_interval = Base\_interval + Rank * Time\_unit \tag{3.1}$$

Base_interval is the smallest DIO interval, which is the DIO interval of the sink. *Rank* stands for the current Rank value of a node. *Time_unit* is the incremental step in the DIO frequency. *DIO_interval* dynamically changes when the Rank of nodes is updated due to changes in the topology.

3.3 RRD+ : Enhanced version of RRD

In reality according to multipath propagation effects (known as fading effects), the received power at certain distance follows a random behavior [50]. Nodes that are close to the edge of transmission range will suffer from frequent Rank updating, which may cause parent nodes loss in parents set. This phenomena is more likely to happen in confined areas with the complexity of the environment makes most of the links instability. Therefore, we proposed RRD+, which is an improvement over RRD that takes into account hysteresis of the coverage zone of the transmission range of nodes. In what follows we will describe in details how RRD+ operates.

3.3.1 Link existence and movement direction monitoring

The same as RRD, RRD+ obtains RSSI values from control messages: broadcast DIO messages and ACK. A node stores two RSSI values for each parent node, Old RSSI value (*OldRSSI*) and New RSSI value (*NewRSSI*). We compare these two values in order to monitor movement direction. We consider that a node is getting closer to its parent if *NewRSSI* is higher than *OldRSSI*. Whereas, we consider that a node is getting further away from its parent if *NewRSSI* is lower than *OldRSSI*.

Due to unpredictable path attenuations, RSSI values might vary even when both nodes do not move. In order to take this phenomenon into account, we introduce

two RSSI thresholds: Safe Threshold ($SAFE\_THRESHOLD$) and Hysteresis Threshold ($HYST\_THRESHOLD$), where $SAFE\_THRESHOLD > HYST\_THRESHOLD$ as shown in figure 3.2. We use a dotted line for $SAFE\_THRESHOLD$ and a dashed line for $HYST\_THRESHOLD$. Note that in our simulation model we used a probabilistic propagation model to take into account coverage zones instability.

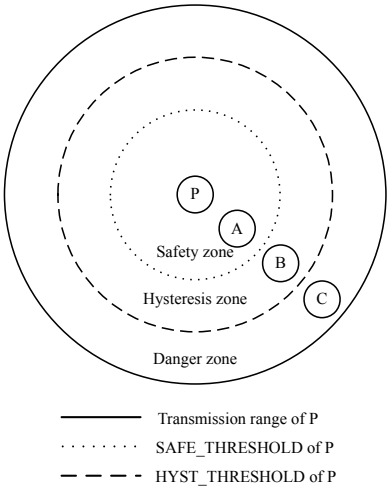


FIGURE 3.2: Node P is the parent node of nodes A, B and C. A is in the Safety zone of node P. B is in the Hysteresis zone of node P. C is in the Danger zone of node P.

When $NewRSSI$ is higher than or equal to $SAFE\_THRESHOLD$, the node is considered to have a good link quality with its parent node; this is the case of node A in figure 3.2. Thus, we consider node A to be in its parent (node P) Safety zone.

When $HYST\_THRESHOLD < NewRSSI \leq SAFE\_THRESHOLD$, which is the case of node B in figure 3.2, we need to detect movement direction first and then consider whether to stop using the link or not. In order to reduce the coverage zone variation influence, we add a hysteresis value to $OldRSSI$ when comparing it to $NewRSSI$. In this situation, we consider node B to be in the Hysteresis zone.

When $NewRSSI$ is smaller than $HYST\_THRESHOLD$, which is the case of node C in figure 3.2, only $NewRSSI$ and $OldRSSI$ will be used to estimate direction without using hysteresis. If node C is getting closer to node P, we consider that node P can still be used as a next-hop. Otherwise, node P should be deleted from the parents set of node C. In this case, we consider node C to be in the Danger zone of node P.

Each time a node receives a DIO message and adds a new node into its parents set, a timer is set for this parent. We call this timer a parent lifetime. Nodes are kept in the parents set and can be selected as next-hops for the lifetime duration. When the timer expires, nodes need to be deleted from the parents set. However, before the timer expires, if a new control message is received from this parent, the lifetime of this parent node will be reset in parents set. According to RRD+ mechanism, we set two sorts of lifetimes: Long Lifetime $Long\_Lifetime$ and Short Lifetime $Short\_Lifetime$.

$Long\_Lifetime$ is given to a parent node in case $NewRSSI$ is higher than $SAFE\_THRESHOLD$. In case $NewRSSI$ is smaller than $SAFE\_THRESHOLD$, movement direction will be estimated first and $Short\_Lifetime$ is given to parent node when movement direction is towards parent node, otherwise Rank value will be updated. Indeed, when the parent node is in Hysteresis or Danger zones where the radio link

is about to fail, *Short_Lifetime* will help avoid using this parent node for a long period in case we no longer receive its control messages. When Rank value is updated, the current parent is removed from the parents set.

3.3.2 Loop avoidance and Rank update

Rank updating is an important mechanism in our proposal. RRD simply uses comparison between *THRESHOLD* and received power to decide Rank updating. However, the received power at certain distance randomly varies due to fading effects in reality. While using RRD, nodes that are close to the edge of transmission range will frequently update Rank value. Frequent Rank updating may cause parent nodes loss in parents set, which easily results in an empty parents set. Therefore, while updating Rank value we take into account hysteresis of the coverage zone of the transmission range of nodes in RRD+, which helps nodes at the edge of transmission range reduce frequency of Rank updating. In what follows we will describe the Rank updating process of RRD+.

With RRD+, nodes monitor links existence and movement direction to allow nodes to update their Ranks in a timely manner. Algorithm 3 depicts the Rank update process. In algorithm 3, *Rank_r* stands for receiver Rank and *Rank_s* stands for sender Rank. *MinIncrease* stands for the value of MinHopRankIncrease.

In figure 3.3, P is the parent node of nodes A, B, C, D and E. Node F is a child node of node B and C. All these nodes will execute algorithm 3 whenever they receive a DIO or ACK message from their parent nodes. Algorithm 3 is explained in what follows and the scenario of figure 3.3 is used as an example to explain Rank updates.

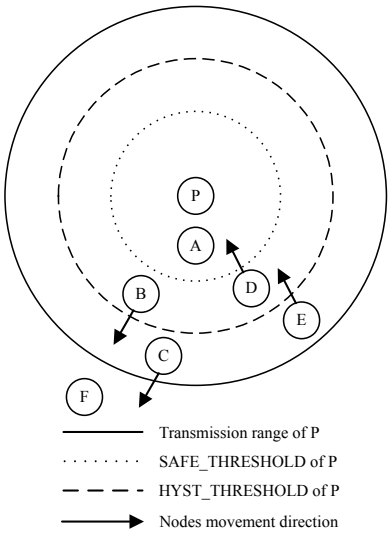


FIGURE 3.3: The position of node A is within *SAFE_THRESHOLD* of node P. Node B and node D are in the Hysteresis zone between *SAFE_THRESHOLD* and *HYST_THRESHOLD* of node P. Node C and node E are between *HYST_THRESHOLD* and the transmission range of node P. Node F is out of the transmission range of node P, but is neighbor of node B and node C.

When a control message is received, if *Rank_r* is higher than *Rank_s*, case of nodes A, B, C, D and E receiving a control message from P, they need to check *NewRSSI* of the control message. If *NewRSSI* is higher than *SAFE_THRESHOLD*, the receiver node sets a *Long_Lifetime* for this parent and updates its Rank $Rank\_r = Rank\_s + MinIncrease$. This is the case of node A.

```
begin
    if received a control message then
        if Rank_r > Rank_s then
            if NewRSSI > SAFE_THRESHOLD then
                Rank_r = Rank_s + MinIncrease;
                Put this parent node in the parents set;
                Set Long_Lifetime for this parent node;
            else if HYST_THRESHOLD < NewRSSI ≤ SAFE_THRESHOLD
             then
                if (NewRSSI − OldRSSI) < HYSTERESIS then
                    Rank_r = Rank_r + MinIncrease;
                    Remove this parent node from the parents set;
                else
                    Rank_r = Rank_s + MinIncrease;
                    Put this parent node in the parents set;
                    Set Short_Lifetime for this parent node;
                end
            else if NewRSSI ≤ HYST_THRESHOLD then
                if NewRSSI ≤ OldRSSI then
                    Rank_r = Rank_r + MinIncrease;
                    Remove this parent node from the parents set;
                else
                    Rank_r = Rank_s + MinIncrease;
                    Put this parent node in the parents set;
                    Set Short_Lifetime for this parent node;
                end
        else
            Ignore
        end
    end
end
```

Algorithm 3: Rank update for nodes in mobility.

If $HYST\_THRESHOLD < NewRSSI \leq SAFE\_THRESHOLD$, case of nodes B and D, in order to avoid the influence of signal fading we set a hysteresis when estimating movement direction with $NewRSSI$ and $OldRSSI$. If $(NewRSSI - OldRSSI) < HYSTERESIS$, case of node B, this means the receiver and the sender are getting away from each other. In this case, the receiver updates its own Rank according to $Rank\_r = Rank\_r + MinIncrease$. When Rank value is updated, children nodes of receiver node will no longer choose it as a parent. This is the case of node F that will no longer consider node B as a parent node once node B has updated its Rank. Indeed, this will help avoid using a node, which might lose its current link with its parent, to be selected as a parent node. In case receiver node moves back towards sender node, it will receive a new DIO message from the sender node and will update its Rank according to the algorithm.

Otherwise, in case $(NewRSSI - OldRSSI) \geq HYSTERESIS$, receiver node gives a short lifetime for the sender node and updates its own Rank according to $Rank\_r = Rank\_s + MinIncrease$. This way, receiver node is only one level below the sender node. This is the case of node D. Receiver node D sets the parent P with a $Short\_Lifetime$ in order to avoid keeping the sender node P as a parent for a long period in case the receiver node D moves away from the sender node P.

If $NewRSSI$ is smaller than or equal to $HYST\_THRESHOLD$, case of nodes C and E, the receivers check the difference between $NewRSSI$ and $OldRSSI$ to estimate movement direction without using hysteresis. In this case the receiver nodes C and E are at the edge of transmission range of the sender node P, thus, updating information fast is imperative, therefore the receivers ignore the hysteresis test. In this situation, if $NewRSSI$ is higher than the old one, case of node E, this means that the receiver is at the edge of transmission range but is moving towards node P. In this case node E gives a $Short\_Lifetime$ for node P and updates its own Rank according to $Rank\_r = Rank\_s + MinIncrease$. Otherwise, if $NewRSSI$ is smaller than the old one, case of node C, this means that receiver node C is getting away from the sender P. In this case, node C increments its own Rank by $Rank\_r = Rank\_r + MinIncrease$. Finally, in case $Rank\_r$ is smaller than $Rank\_s$ we ignore the control messages.

3.4 MRRD+ : Mobility enhancement for RPL with multiple sinks

In a mobile single-sink scenario all nodes except the sink are free to move and transmit data to the sink in a continuous manner. However, only one sink in the network will make paths from nodes to the sink as they move further away from the sink. Adding additional sinks in the network would reduce the number of the hops between source nodes and the destinations. Besides, in a convergecast WSN only one sink easily leads to a faster energy depletion of the sink, more packet loss, higher latency and smaller network range. Deploying multiple sinks in the network can help solve these problems. Therefore, we propose MRRD+, which is an enhancement over RRD+ that takes into account multiple sinks that share the same DODAGID (See Chapter 2 section 2.4). Besides, we improved dynamic control message management in order to inform Rank updates as soon as possible in the case where a node is trying to move out of its parent node range.

3.4.1 Common techniques between RRD+ and MRRD+

MRRD+ follows approaches used in RRD+ to monitor link existence and update Rank. Since all the sinks share the same DODAGID, the algorithm 3 is applied with multiple sinks . Nodes will consider all potential parent nodes with the same Rank value regardless of the sink nodes, and the Rank update stays the same.

3.4.2 Enhanced ripple control message management

In 3.2.2 we introduce ripple control message management, and it dynamically modifies the frequency of DIOs according to Rank updates and allows each node to have its own frequency for DIO messages by the equation $DIO\_interval = Base\_interval + Rank * Time\_unit$. However, when considering the coverage zone we defined, if a node is in Hysteresis zone or Danger zone and moving away from its parent node, the Rank value will increase according to algorithm 3. It is essential to inform its neighbors of the updated Rank value. Therefore, in this case DIO should be sent frequently. If the node is in Safety zone or moving towards its parent node, the Rank value will not be increased. Frequent DIO messages are not necessary. Therefore, in this case we need to reduce DIO frequency. Based on this, we propose algorithm 4 to enhance dynamic DIO interval management.

Initialize:
$a = 0$;
$DIO\_interval = Base\_interval + Rank * Time\_unit$;
begin
  if *DIO is received* **then**
    if $NewRank \neq OldRank$ **then**
      $a = 0$;
      $DIO\_interval = Base\_interval$;
    else
      $a = a + MinIncrease$;
      $DIO\_interval = Base\_interval + a * Time\_unit$;
      if $DIO\_interval > DIO\_max$ **then**
        $DIO\_interval = DIO\_max$;
      end
    end
  end
end

Algorithm 4: Dynamic DIO interval management.

In algorithm 4, each node would set its DIO interval to $Base\_interval + Rank * Time\_unit$ during the network construction. $Base\_interval$ is the smallest DIO interval, which is the DIO interval of the sink. $Rank$ stands for the current Rank value of a node. $Time\_unit$ is the incremental step in the DIO frequency. $NewRank$ stands for the Rank value after a Rank update according to algorithm 3 when receiving a control packet from other nodes. $OldRank$ stands for the previous Rank value. If a node changes its Rank value ($NewRank \neq OldRank$), this means that the node is out of the Safety zone and moving away from its parent node, or the node receives control messages from a node which has a lower Rank value than its current parent node. In this case, the node needs to broadcast DIO messages more frequently. Therefore, the $DIO\_interval$ will be set to $Base\_interval$ at first, which is the minimum $DIO\_interval$. After $Base\_interval$, if this node keeps the same Rank value, it sets

$DIO\_interval$ to $Base\_interval + a * Time\_unit$, where a increases $MinIncrease$ each time, in order to reduce the cost of overhead. Assuming that the maximum Rank value of the network is Max_Rank. According to this equation, we could get the following maximum DIO interval $DIO\_max = Base\_interval + Max\_Rank * Time\_unit$. When $DIO\_interval$ is larger than DIO_max, it stops increasing $DIO\_interval$.

3.5 ADUP : Mobility enhancement mechanism for convergecast and anycast WSNs

ADUP is an adaptive routing protocol that integrates anycast and convergecast traffic together and supports highly mobile scenarios. All nodes except the sink are free to move and need to send periodic data to the sink. Simultaneously, the sink needs to periodically send command packets to randomly selected nodes in the network. This protocol reacts quickly to the movement of nodes to make faster decisions for the next-hop selection for convergecast traffic and dynamically build routes for anycast traffic. Note that mobility support of convergecast is achieved according to RRD+.

3.5.1 Building dynamic next-hop table during upward routing

In RRD+, all neighbors with lower Ranks form a set that we call parents set. In a data collection process, before sending a packet, a node needs to select a next-hop from its parents set. In ADUP, the ID of this next-hop is included into data packets and sent to the sink. Instead of storing the entire addresses of next-hop nodes in data packets, we only use 1 byte to store the ID of the next-hop node of the source node. Due to the fact that the upper limit value of 1 byte is 255, the maximum number of nodes in the network cannot exceed 255. Figure 3.4 shows the fields of upward data packets. When the sink receives data packets, it builds a next-hop table as shown in figure 3.5. The first column of this table stands for the ID of nodes, except the sink, in the network. We consider that there are n nodes and one sink in the topology. We define the ID of each node as $ID_i \in \{ID_0, ID_1, ID_2, ..., ID_n\}$, where ID_0 stands for the ID of sink. The second column stands for the ID of best next-hop for each node referred to as $N(ID_i)$. Note that $N(X) \in \{ID_0, ID_1, ID_2, ..., ID_n\}$ and $X \in \{ID_1, ID_2, ..., ID_n\}$.

128 Bytes					
0-29	30-31	32-33	34	35-36	37-127
Data	Sender address	Receiver address	ID of next-hop	Packet Id	Idle

FIGURE 3.4: The fields of upward data packets.

RRD+ updates the Rank value of each node according to link quality and movement direction. Nodes in a parents set will be automatically removed or added based on the variation of Rank values. Thus, the next-hop of each node will also dynamically change according to movement. Due to the fact that each node periodically sends packets to the sink, the next-hop table will adapt to mobility periodically.

ID	Nexthop
ID_1	$N(ID_1)$
ID_2	$N(ID_2)$
...	...
ID_n	$N(ID_n)$

FIGURE 3.5: Dynamic next-hop table of the sink node.

3.5.2 Building downward routing route

In order to reach the destination through multiple hops, the sink needs to build a route before sending a packet. Algorithm 5 depicts the route building process. We use ID_d to stand for the ID of the destination node which will be put into the route first. The sink will extract the preferred next-hop of ID_d from the dynamic next-hop table. If $N(ID_d)$ equals ID_0, this means that the sink can reach node ID_d directly and the building process stops immediately. In case $N(ID_d)$ is not ID_0, the sink continues the route building process. For any entry in the first column of dynamic next-hop table, the entry ID_i that equals $N(ID_d)$ will be put into the route and its next-hop $N(ID_i)$ needs to be compared to ID_0. The building process will stop when the next-hop of item ID_i is ID_0. The building process of a route is done from the destination to the sink, the route we get is in reverse order and it is thus reversed before it is used as a path.

Input: ID_d
Output: *Route*
begin
  Put ID_d in the *Route*;
  $Nexthop = N(ID_d)$;
  while *Nexthop does not equal to* ID_0 **do**
    for *each item i in* ID_i **do**
      if $ID_i = Nexthop$ **then**
        $Nexthop = N(ID_i)$;
        Put ID_i in the *Route*;
      end
    end
  end
  Reverse(*Route*);
end

Algorithm 5: Route building.

Figure 3.6 shows how algorithm 5 works. ID_d is put into the route first. During this process, numbers of ID_i will be put in the route until the next-hop of ID_x is found to be ID_0. At the end, the route needs to be reversed.

We consider that there are m nodes in the route. Before sending a data packet, the sink needs to store the IDs of these nodes in the data packet as shown in figure 3.7. Every time the data packet is relayed, the route offset will be increased to help relay nodes to find the next-hop within m bytes until reaching the destination.

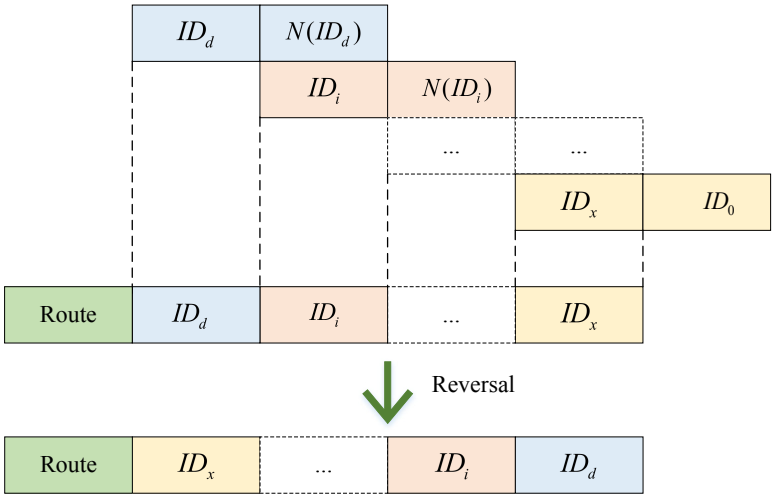


FIGURE 3.6: Route building process.

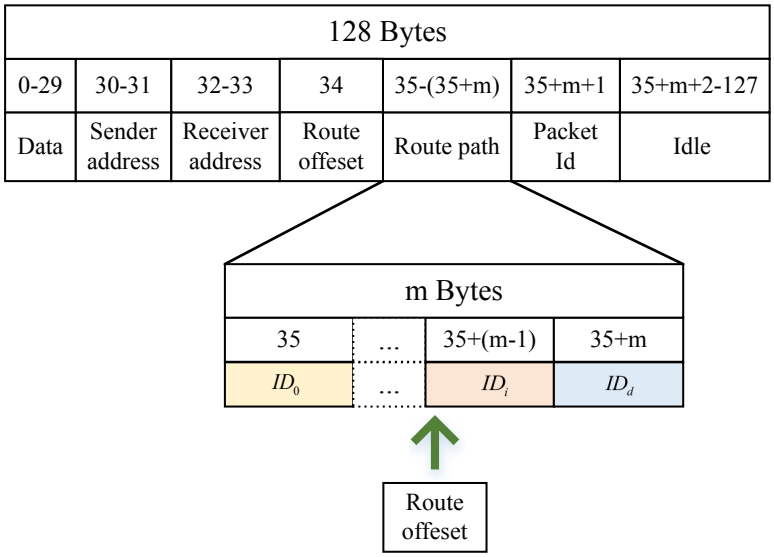


FIGURE 3.7: The fields of downward data packets.

3.6 Summary

Introducing mobility in LLNs is not a straightforward task. The limited resources of nodes makes it very challenging to find optimized solution. In this chapter, we present RL, RRD, RRD+ routing protocols to support mobility in highly mobile convergecast scenarios. RL is a mechanism that can be used in most routing protocols to support mobility. However, due to periodically broadcasting control messages RL suffers from high overhead. When running RL over RPL, we propose ripple control message management in RRD to enhance its performance on overhead. RRD+ is an updated version over RRD that takes into account hysteresis of the coverage zone of the transmission range of nodes. Evaluation results presented in chapter 4 shown the efficiency of RRD+ compared to some existing protocols.

MRRD+ is an enhancement over RRD+ that takes into account multiple sinks that share the same DODAGID. We improve ripple control message management in order to inform Rank updates as soon as possible in the case where a node is trying to move out of its parent node range.

Based on RRD+, we propose ADUP to support mobility in highly mobile convergecast and anycast scenarios. Convergecast part is supported by RRD+. This protocol reacts quickly to the movement of nodes to make faster decisions for the next-hop selection in data collection and dynamically build routes for anycast traffic. Results in chapter 4 through simulation show that our work outperforms two generic ad hoc routing protocols AODV and flooding on different performance metrics.

Chapter 4

Results

In this chapter, we evaluate the performance of our contributions through simulation and experiment (only ADUP is experimented). The simulation is by using Contiki operating system and its simulator Cooja. This chapter is organized as followed. The first section 4.1 introduces simulation system, path loss of propagation, mobility model used for simulation and performance metrics for evaluation. In the second section 4.2, we show the efficiency of RL mechanism while using different link metrics, and evaluate the performance of RL and RRD when they are implemented on top of RPL. In section 4.3, we compare RRD+ with other existing mobility support methods mentioned in Chapter 2 on different degrees of mobility. Section 4.4 presents the performance of MRRD+ in multi-sink scenarios. In section 4.5 we present ADUP simulation results and compare it with two routing protocols. In section 4.6 we present ADUP experimental results and compare it to standard RPL. Finally section 4.7 summaries this chapter.

4.1 Simulation system

Contiki is a lightweight open source operating system for tiny network devices including sensor nodes [51]. It is designed to run on the hardware devices that are constrained in memory, communication bandwidth and processing power. Contiki is developed in C, therefore it is highly portable to different architectures like Texas Instruments MSP430. Contiki is an event-driven system in which processes are implemented as event handlers. Contiki provides three basic network mechanisms: the μIP (micro IP) TCP/IP (Transmission Control Protocol/Internet Protocol) stack with IPv4 (Internet Protocol version 4) networking, the μIPv6 stack with IPv6 networking, and the Rime stack. The IPv6 stack contains RPL routing protocol, the 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) header compression and adaptation layer for IEEE 802.15.4. Rime is a set of lightweight networking protocols designed for low-power wireless networks. The Rime stack provides a set of communication primitives. The primitives can be used on their own or combined to form different protocols.

4.1.1 Path loss of propagation

The Contiki system includes a network simulator called Cooja. The simulator is implemented in Java but allows the software of sensor nodes to be written in C. Compared with many existing network simulators such as NS2, NS3, OMNeT++, TOSSIM, etc, Cooja is more compatible with real hardware and the simulation code can be used on real life nodes directly. Besides, we made some modifications on the default parameters of Cooja in order to make it more realistic, especially when it comes to emulating signal propagation. There are four propagation models in Cooja

[52]. One of them is Multi-path Ray-tracer Medium (MRM) which takes reflections and refractions into account to simulate real environment. The MRM radio model relies on ray tracing technology and integrates capture effect[1].

In order to make it more suitable for unstable environments, such as mobile networks deployed in confined spaces, we included a random behavior to path loss calculation. Indeed, we increase path loss exponent to take into account the presence of obstacle between two nodes and add Gaussian random variable in the path loss formula of MRM in order to simulate instability of the radio links. The formula of the received signal power in decibel (dB) is presented in equation 4.1:

$$PL(d) = PL(d_o) + 10\alpha log_{10}(\frac{d}{d_o}) + X_\sigma \tag{4.1}$$

Where $PL(d)$ is the total path loss measured in dB and $PL(d_o)$ is the path loss at the reference distance d_o in dB. d is the length of the path and d_o is the reference distance. α is the path loss exponent, which must be measured at the site of the planned deployment. Existing works show that the path loss exponent value α range between 2.7 and 3.5 for an urban area [53] and we set it to 3 to simulate transmission range around 40 meters with $-20\,dB$ transmission prower in Cooja. X_σ is a Gaussian random variable in the interval $[-2, 2]$ with mean value of 0 and a standard deviation σ of 1. The random component with a Gaussian standard deviation makes each node transmission range change from around 30 meters to 50 meters randomly.

The modified MRM ration propagation model with its new feature is used to evaluate our contribution.

4.1.2 Mobility model

Mobility models are used for simulation when network protocols are evaluated. In order to simulate the movement of mobile sensor nodes, it is necessary to select suitable mobility models for the specific application. There are two types of mobility models: individual mobility and group mobility. In the first type model, the movement of nodes is independent of each other; the common models are Random Waypoint, Random Direction and Random Walk [54]. Random Direction and the Random Walk are two variants of the Random Waypoint. In the second type model, mobile nodes are divided into different groups and the nodes in one group follow the same specific movement. This type model is usually used in military environments, since soldiers move in group [55].

In our work we assume that mobile nodes move randomly and freely without restrictions. Thus, we select Random Waypoint as our simulation mobility model. Random Waypoint is a commonly used model in Ad Hoc networks. In this model each node begins by pausing for a fixed number of seconds. The mobile node then selects a random destination in the simulation area and a random speed between 0 and maximum speed. The node moves to this destination and again pauses for a fixed period before another random location and speed. In our simulation all nodes are randomly deployed in a 200 m * 200 m area at the beginning. The sink is always located at the center of the area for the duration of the simulation. When the simulation starts, each node makes a random choice to select a location within the area as a destination. Then, it moves towards this destination. The velocity is randomly chosen from [1 m/s, 3 m/s] every 5 seconds, which is a simulation of walking speed of human in a building. We introduce rest time for each node. Whenever a node

[1]The capture effect is a phenomenon associated with frequency modulation reception in which only the stronger signals with a certain threshold will be demodulated.

TABLE 4.1: Mobility model parameters.

Item	Parameters
Mobility Model	Random Waypoint Model
Minimum speed	1 m/s
Maximum speed	3 m/s
Speed changing interval	5 s
New location rest time	5 s

reaches its destination position, it will rest in new location for 5 seconds and then selects a new destination for the next movement. We introduce 5 seconds stop in order to simulate human walking stop before selecting the new destination. All nodes, except the sink, repeat these steps until the simulation stops. Table 4.1 summarizes mobility model parameters for simulation.

4.1.3 Unslotted CSMA/CA

The IEEE 802.15.4 unslotted CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) aims to reduce interferences and collisions by sensing the channel and waiting for random periods before accessing the medium. The main part of this algorithm is described in the following paragraph.

Each node maintains two variables for each transmission attempt: NB (number of backoffs) and BE (backoff exponent). NB is the number of times that CSMA/CA algorithm was required to backoff while attempting the current transmission. NB is initialized to zero before each new transmission attempt. BE is the backoff exponent, which is the backoff periods that a node shall wait before attempting to assess the channel. BE shall be initialized to the value $macMinBE$. Any transmission is delayed for a random number of backoff periods in the range $[0; 2^{BE} - 1]$. After this delay, channel sensing is performed by doing clear channel assessment (CCA). If the channel is busy, both NB and BE are incremented by one, ensuring that BE shall be no more than macMaxBE. If NB is less than or equal to macMaxCSMABackoff, the algorithm shall return to step of setting backoff periods. In case NB reaches macMaxCSMABackoff, the algorithm shall terminate, which means that the node does not succeed in accessing the channel. If the channel is assessed to be idle, the CSMA/CA succeed and the transmission may start.

The MAC layer of Contiki OS provides a version of the CSMA/CA algorithm. Indeed, the provided version does not respect the backoff procedure and does not use the backoff periods specified in the standard. To overcome this insufficiency, we implemented a compliant version of IEEE 802.15.4 unslotted CSMA/CA in Contiki 3.0. We used this compliant version of CSMA/CA to evaluate our contributions.

4.1.4 Performance metrics

We use seven performance metrics to evaluate the efficiency of our work: (i) packet delivery ratio, (ii) number of dropped packets, (iii) dropped packets ratio, (iv) number of retransmissions, (v) number of sent packets, (vi) average end-to-end delay and (vii) number of control packets.

Packet delivery ratio is defined as the ratio of received data packets at the sink to those generated by the source nodes. The packet delivery ratio metric shows the

ability of the routing protocol in successfully delivering packets to the final destination. This evaluation metric is very important in showing how well each variant is able to cope with mobility.

According to IEEE 802.15.4 MAC layer, nodes keep retransmitting the same frame until the number of retransmission attempts reaches a fixed maximum value after which the frame is dropped. The number of maximum retransmissions is 4 according to the standard. The number of dropped packets is the number of packets that are dropped after exceeding the maximum number of retransmission attempts. This evaluation metric shows how efficient the routing protocol is in finding a new parent when the link with the current parent is lost. Dropped packets ratio is the ratio between the number of dropped packets and the number of generated packets.

Each node may send the same packet 5 times due to collisions and link failures. The number of retransmissions gives an idea about the congestion of the network and efficiency in avoiding to overload the network with retransmissions.

Total number of sent packets refers to the number of packets that are sent over the medium during the simulation. Nodes in our simulations generate the same number of packets but each node may send a different number of frames depending on the retransmission attempts due to collisions and packet loss. This performance metric gives us the total number of packets that each node sent for the same number of generated packets. An efficient metric should have fewer sent packets for the same number of generated and received packets.

The end-to-end delay refers to the duration taken by a packet from a node to the sink. Measuring the end-to-end delay helps to show the ability of the routing protocol in building efficient routes with the fewest number of hops to reach the destination and its ability to avoid routing loops that will increase this delay. To compute the average end-to-end delay we measure the delay taken by total received packets and divide it by the number of successful received packets at the sink. The packet average end-to-end delay can be computed using the following two equations.

$$Total\ Delay = \sum_{i=1}^{n} (RecvTime(i) - SentTime(i)) \tag{4.2}$$

$$Average\ end\ to\ end\ Delay = Total\ Delay/n \tag{4.3}$$

Where n stands for the number of successfully received packets.

Lastly, we evaluated the overhead of each enhancement in order to show the additional work done by the protocol. This gives an idea about the complexity of the enhancement in terms of number of control messages and in a certain extent, it gives an idea about the additional energy consumption that it induces.

4.2 RL and RRD performance evaluation

We chose to show the efficiency of our RL and RRD mechanism using three of the most used metrics: number of hops (HopCount), latency, and expected transmission count (ETX). HopCount is a metric that is based on the number of hops that separates source node from destination (sink node). Nodes select the node that has the smallest hop count in the neighbors set or parents set as their next-hop. For the latency metric, we calculate the the queuing delay, which is the time spent by a packet in the packet queue of the MAC layer. The node computes the difference between the queuing time and dequeuing time of a packet in the queue. We make an average calculation of the last ten packets as it is done in [56]. If the number of dequeued

packets is less than ten, the average calculation is only computed over the number of dequeued packets. Nodes choose the next-hop based on the smallest path delay, which represents the estimated time needed to reach the sink based on the queuing delay of intermediate nodes. For the ETX, we use an average value of last ten packets as ETX metric and select the next-hop based on the smallest path ETX in the neighbors set or parents set in a similar way to what is done in [57].

We applied HopCount, latency and ETX with RL mechanism and we call these new combinations as RL-HopCount, RL-Latency and RL-ETX. Unlike the latency and ETX, HopCount is a quantized metric. Normally in the neighbors set or parents set many nodes have the same smallest number of hops to the sink. Using HopCount metric, we always select the first node in the neighbors set or parents set with the smallest number of hops. In addition to these 3 metrics, we also introduce a random variant method which consists on randomly choosing the next-hop from the nodes that have the same smallest hop count. We call this metric Random and its implementation with RL mechanism RL-Random. We also applied RL mechanism to RPL. Since the native RPL does not introduce Random as its metric, we only included RL to RPL-Hopcount, RPL-ETX and RPL-Latency, and we called them RL-RPL-Hopcount, RL-RPL-ETX and RL-RPL-Latency. RRD is an enhanced version of RL and proposed for RPL. Thus we only evaluate the performance of RRD when it is implemented in RPL, and we called them RRD-RPL-Hopcount, RRD-RPL-ETX and RRD-RPL-Latency.

Simulations are performed using IEEE 802.15.4 MAC. Nodes access the medium using unslotted CSMA/CA algorithm. We use a packet queue length of 16 packets with a 30 byte packet size. We disable the duty cycling to keep nodes awake all the time. In order to test our work in different network densities, we continuously add 20 nodes from 20 to 60 with one sink. According to RPL specification [13], there are two parameters we need to set for default RPL when using trickle algorithm. $DEFAULT\_DIO\_INTERVAL\_MIN$ is the default value used to configure I_min in DIO trickle timer. We set this value to 11 which results in I_min of 2 seconds. The default value used to configure I_max in DIO trickle timer is $DEFAULT\_DIO\_INTERVAL\_DOUBLINGS$. We set this value to 1. This configuration results in a maximum interval of 4 seconds. These configurations are decided after multiple simulation tests with different value combinations of $DEFAULT\_DIO\_INTERVAL\_MIN$ and $DEFAULT\_DIO\_INTERVAL\_DOUBLINGS$. We found out that RPL copes with mobility fast with $DEFAULT\_DIO\_INTERVAL\_MIN$ equaling 11 and $DEFAULT\_DIO\_INTERVAL\_DOUBLINGS$ equaling 1. Compared to default settings $DEFAULT\_DIO\_INTERVAL\_MIN = 3$ (8 millisecond) and $DEFAULT\_DIO\_INT\text{-}ERVAL\_DOUBLINGS = 20$ (2.3 hours) indicated by the specifications. Our configuration increases I_min, but decreases I_max, which allows nodes to better cope with topology changes in mobility. Table 4.2 summarizes the rest of the simulation parameters we used.

We used five performance metrics to evaluate the enhancement of RL and RRD: (i) packet delivery ratio, (ii) number of sent packets, (iii) number of retransmissions, (iv) number of dropped packets and (v) average end-to-end delay. For each network size, we generated 10 different random mobility scenarios. Each performance metric is averaged over 10 iterations for each network size.

TABLE 4.2: Simulations parameters used to evaluate RL and RRD performance.

Item	**Parameters**
Network simulator	Cooja under contiki OS (3.0)
Radio propagation model	MRM with random behavior
Medium access control	CSMA/CA
Simulation time	5 minutes
Emulated platform	Z1 starter platform
Sensor Nodes Deployment	Random Deployment
Data size	30 Bytes
Packet queue size	16
Transmission rate	1 pkt/sec
Transmission power	-20 dBm
Transmission range	[30 m, 50 m]
Number of nodes	20, 40, 60
Area of deployment	200 m x 200 m
Frequency range	2.4 GHz
Receive Sensitivity	-95 dBm
RSSI Threshold $THRESHOLD$	-92 dBm
Long lifetime $Long\_Lifetime$	40 s
Short lifetime $Short\_Lifetime$	25 s
$DEFAULT\_DIO\_INTERVAL\_MIN$	11
$DEFAULT\_DIO\_INTERVAL\_DOUBLINGS$	1
Number of iterations	10

4.2.1 Packet delivery ratio

Figure 4.1 shows that Random, HopCount and ETX outperform Latency, which is mainly due to the fact that Latency has more loops and collisions during the transmission. All four metrics have better packet delivery ratio when used with RL mechanism. RL-Random performs better than the other variants mainly due to the fact that the random choice in parents set balances the traffic load. Indeed, sending packets to a different parent node at each transmission increases the probability of success of packet delivery and decreases the probability of sending packets to a node that is moving away. Having more nodes in the network increases the traffic load. This may cause network collisions, however, this also helps the parents set contains more potential next-hop options to load balance the network. Therefore, RL-Random has almost the same packet delivery ratio with 40 nodes and 60 nodes.

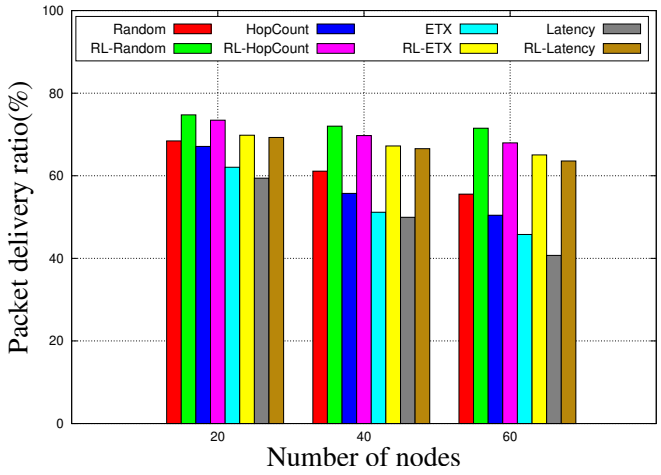


FIGURE 4.1: Packet delivery ratio (RL with different link metrics).

Figure 4.2 shows the packet delivery ratio results for RPL, RL-RPL and RRD-RPL. With RPL, delivery ratio of Latency metric does not decrease much with the increase of number of nodes compared to the routing based on latency presented in figure 4.1. This is mainly due to the rank mechanism in RPL that can help nodes avoid some loops compared with the routing protocols that do not have rank mechanism. The results show that all the three variants with RL and RRD outperform native RPL. This is mainly due to the fact that Rank update method in RL and RRD helps nodes quickly find the suitable parent nodes and avoid packet loss. Note that RRD outperforms RL. RRD enhances RPL more on packet delivery ratio with 60 nodes compared with RL. This is mainly due to the fact RL uses periodical control messages broadcasting, which results in high overhead. However, the ripple control message management helps RRD update topology with less overhead (see subsection 4.2.6 figure 4.11) and further reduce the packets loss.

4.2.2 Number of sent packets

Figure 4.3 shows that Random and HopCount send less packets than ETX and Latency. This is mainly due to the fact that ETX and Latency select the next-hop with best ETX and Latency values, which might lead in some cases to choosing a longer path. The number of sent packets depends on the length of the path that packets followed to reach the sink. Results show that all four metrics send less packets when used with the RL mechanism, because the level setting in RL mechanism helps make

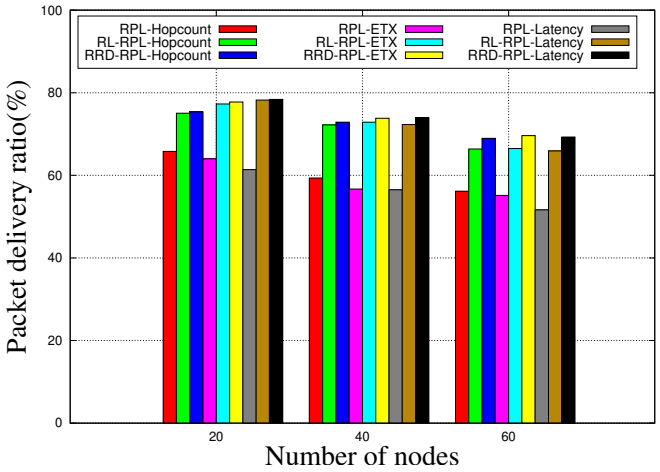


FIGURE 4.2: Packet delivery ratio (RPL, RL-RPL and RRD-RPL).

sure the data packets being transmitted upwards from leaf nodes to the sink which in turn helps reduce the number of transmission hops traveled by packets. Random and HopCount have almost the same number of sent packets with and without RL mechanism with 20 nodes and 40 nodes scenarios, this is essentially due to the fact that they use a metric based on the shortest path and inherently avoid loops. Whereas we can see that RL mechanism enhances ETX and Latency because it helps update the next-hop in a timely manner. In four variants, RL-Random outperforms other variants. This is mainly thanks to the random choices in parents set as explained in the packet delivery ratio results.

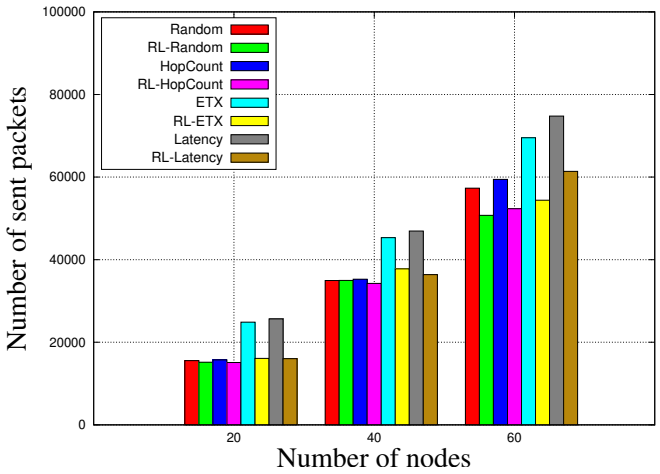


FIGURE 4.3: Number of sent packets (RL with different link metrics).

Figure 4.4 shows that with 20 and 40 nodes the performances are almost the same, but when the number of nodes increases to 60, RL and RRD variants in RPL outperform native RPL. This is mainly due to the fact that in small network the probability of loops with mobility (We discussed loops problem with mobility in Chapter 2 section 2.2.) is smaller and the Rank mechanism of RPL can still help to avoid them. Whereas with 60 nodes the probability of loops with mobility increases and the Rank mechanism is not able to cope with it. Furthermore, figure 4.2 shows a 10 % increase in packet delivery ratio. With 60 nodes RPL-Latency sends more packets compared with RPL-ETX and RPL-Hopcount, which is mainly due to the

fact that the metric of latency will dramatically increase. This easily causes a longer path selection, when there are many collisions (See figure 4.6) in the network. Also note that RL and RRD help reduce the number of sent packets for all metrics in figures 4.3 and 4.4. This shows the efficiency of RL and RRD in sending data packets upwards from leaf nodes to the sink and in finding shorter paths.

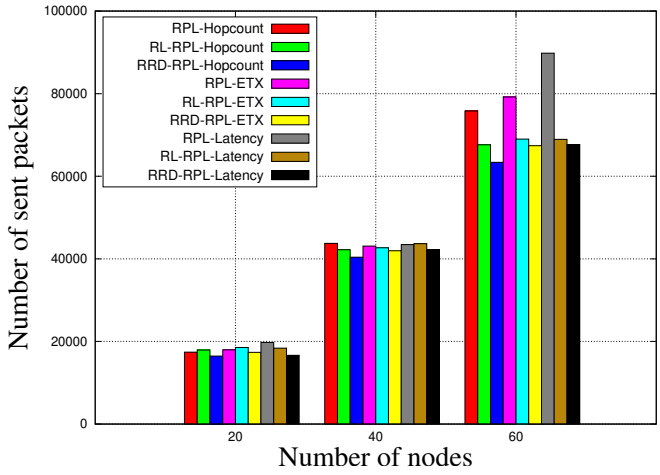


FIGURE 4.4: Number of sent packets (RPL, RL-RPL and RRD-RPL).

4.2.3 Number of retransmissions

Figure 4.5 shows RL variants have better performance when it comes to the number of retransmissions. This is mainly due to the fact that the RL mechanism helps reduce loops in a network which we already discussed in subsection 4.2.2. Fewer loops means fewer transmission hops to the sink, which in turn will reduce the collisions in the network. The Latency metric sends much more packets than other metrics. This is mainly due to the time needed for Latency to update its metric value. A node will wait for new latency values to update its own latency value by doing an average of the past values. This takes too long before and become outdated very quickly because the topology keeps changing.

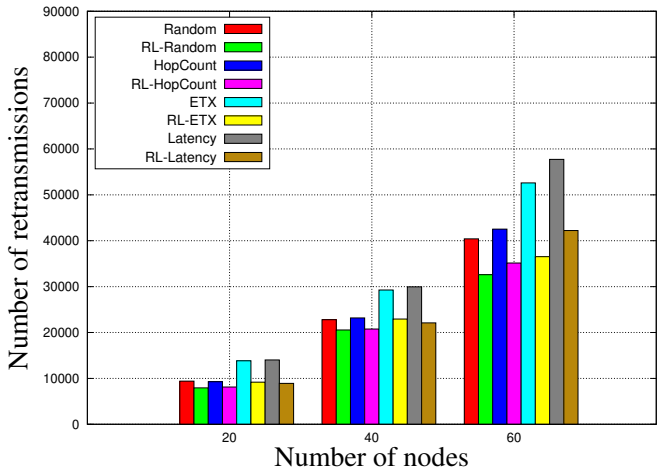


FIGURE 4.5: Number of retransmissions (RL with different link metrics).

Figure 4.6 shows that all the RPL metrics and their variants have almost the number of retransmissions with 20 and 40 nodes. However, when the number of nodes increases to 60, all the RL and RRD variants in RPL outperform the native RPL metrics. Although the native RPL metrics ETX and Latency have almost the same performance in packet delivery ratio with 60 nodes, Latency needs more transmissions to achieve it due to its slow convergence perspective.

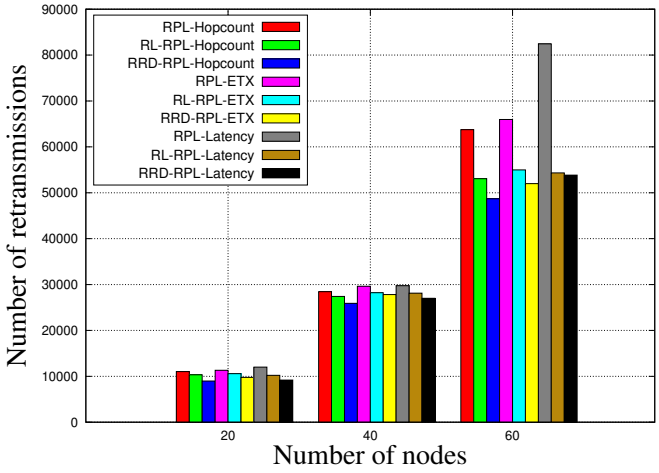


FIGURE 4.6: Number of retransmissions (RPL, RL-RPL and RRD-RPL).

4.2.4 Number of dropped packets

Counting the number of dropped packets in the network helps us analyse the network congestion. Figure 4.7 shows RL variants have fewer dropped packets than the standard routing protocol with different metrics, which means fewer collisions and loops. With the number of nodes increasing from 20 nodes to 40 nodes, the number of dropped packets increases as well. This is due to more nodes will cause more network transmissions. But when the number of nodes increases to 60 nodes, all routing metrics and RL variants do not have a huge increase in the number of dropped packets. This is mainly because more number of nodes gives every node more choices to select next hop and this can efficiently reduce collisions in the network. Latency has a much higher number of dropped packets compared to other metrics, which is mainly due to the fact that the metric of Latency is very depend on the stability of the network. With the movement of the nodes, the value of metric will fluctuate greatly, which makes Latency easily to select a faraway node to be the next-hop.

Figure 4.8 shows that all the RL and RRD variants in RPL outperform the native RPL. This is due to the fact that RL and RRD mechanism help node select optimal path to the sink, which has fewer loops and less collisions.

4.2.5 Average end-to-end delay

Figure 4.9 shows that except Latency, the other metrics with and without RL mechanism have very similar average delay with 20 nodes and 40 nodes. When the number of nodes increases to 60, RL variants show their enhancement in network delay performance. This is mainly due to the fact that with 20 and 40 nodes there are not so many and retransmissions. When the number of nodes is 60 and the topology

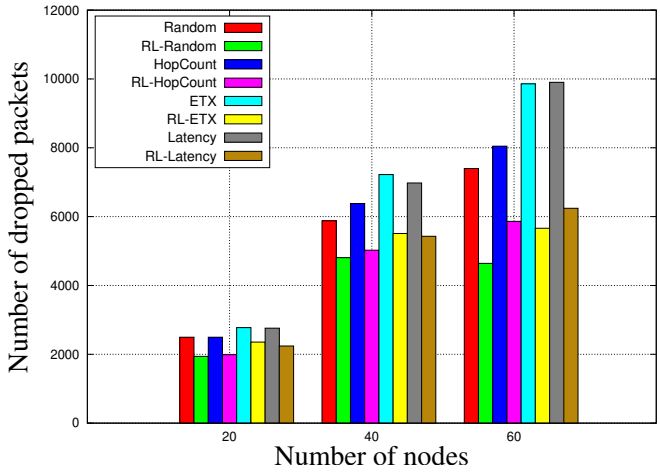


FIGURE 4.7: Number of dropped packets (RL with different link metrics).

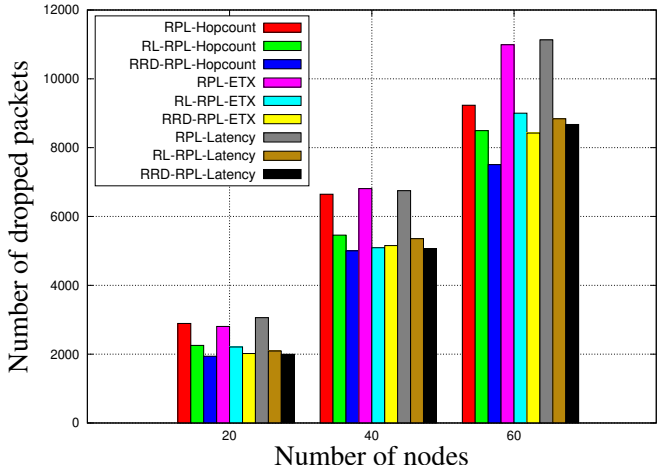


FIGURE 4.8: Number of dropped packets (RPL, RL-RPL and RRD-RPL).

of network becomes more complex, the performance of delay will begin to make difference. Latency shows much higher average end to end delay for all scenarios mainly because the routing metric of Latency is not being updated fast enough to cope with the topology changes as explained in subsection 4.2.3. In addition, Latency suffers from a high retransmission attempts using CSMA/CA. Indeed, after each unsuccessful transmission attempt, a packet will wait for a backoff time to be sent again, which will cause huge delay for a packet. Note that RL-Random outperforms other RL variants due to the random choice for the parent with small number of retransmission attempts.

Figure 4.10 shows that all the metrics with and without RL and RRD mechanism have very similar average delay with 20 and 40 nodes. RL and RRD variants only show their improvement when the number of nodes is 60. This is mainly due to the fact that rank mechanism activates its function with small sized network in mobility. Note that comparing with RL, RDD increases the performance of average end-to-end delay when the number of nodes changes from 40 to 60. This is essentially due to the result of the increase of RL in the number of sent packets, number of retransmissions, and the number of dropped packets. All these factors contribute to the increase of

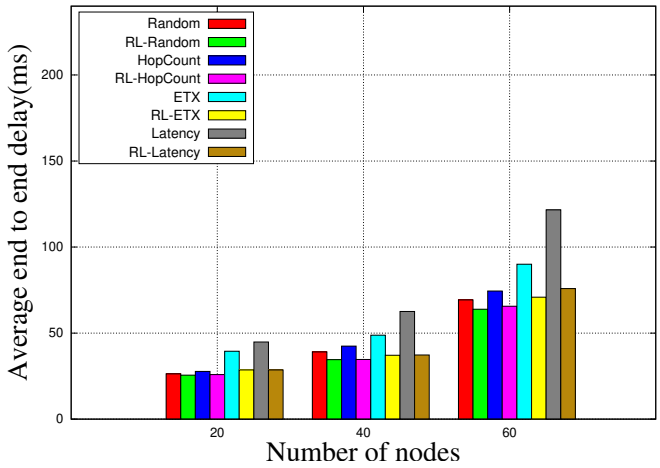


FIGURE 4.9: End-to-end delay (RL with different link metrics).

RL in the end-to-end delay.

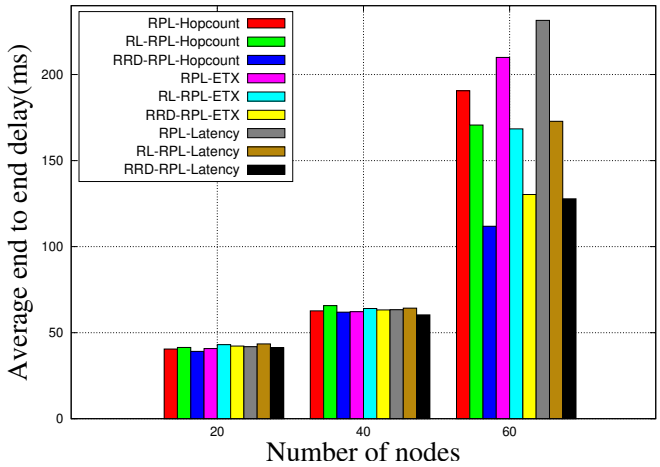


FIGURE 4.10: End-to-end delay (RPL, RL-RPL and RRD-RPL).

4.2.6 Number of control packets

Since RL uses periodical control messages broadcasting, all the RL variants with different link metrics have the same number of control packets. It dose not make sense to discuss number of control packets among different RL variants. Thus, we only discuss the number of control packets of RPL, RL-RPL and RRD-RPL. Control packets are DIO, DIS, DAO and DAO-ACK. Figure 4.11 shows that all RL-RPL variants have almost the same number of control message in different size of network. This is mainly due to the fact that the nodes using RL-RPL periodically broadcast control messages with same period. The figure also shows that RRD variants outperform standard RPL with different metrics and highly reduce the number of control packets for denser networks. This is mainly due to the fact when nodes move away from the sink, ripple control message management in RRD helps enlarge the sending interval, which can reduce the number of DIOs.

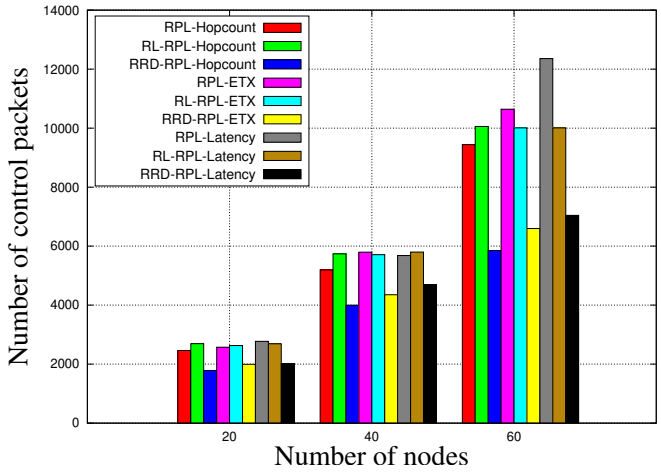


FIGURE 4.11: Number of control packets.

4.3 RRD+ performance evaluation

In order to assess the efficiency of RRD+ in dealing with mobility we compared it to other existing methods studied in the related work section. These protocols are OR-RPL (original RPL), Co-RPL [43], ME-RPL [31], and RT-RPL (Reverse trickle timer algorithm) [30]. We chose 3 sizes for the network: 20 nodes, 40 nodes, and 60 nodes. In addition, we considered different mobility scenarios where the ratio of mobile nodes to fixed nodes in the network are 25 %, 50 %, 75 %, 100 %. For RPL, we set *DEFAULT_DIO_INTERVAL_MIN* to 13 which results in *I_min* of 8 seconds and set *DEFAULT_DIO_INTERVAL_DOUBLINGS* value to 2 which results in *I_max* of 32 seconds. These configurations are decided after multiple simulation tests with different value combinations of *DEFAULT_DIO_INTERVAL_MIN* and *DEFAULT_DIO_INTERVAL_DOUBLINGS*. We found out with this configurations RPL would give attention to mobility detection and overhead saving meanwhile. Table 4.2 summarizes the rest of the simulation parameters we used.

We retained four performance metrics to evaluate the efficiency of RRD+ used on top of RPL compared to other routing protocols: (i) packet delivery ratio, (ii) number of dropped packets, (iii) average end-to-end delay, and (iv) number of control packets. For each network size, we generated 10 different random mobility scenarios. Each performance metric is averaged over 10 iterations for each network size.

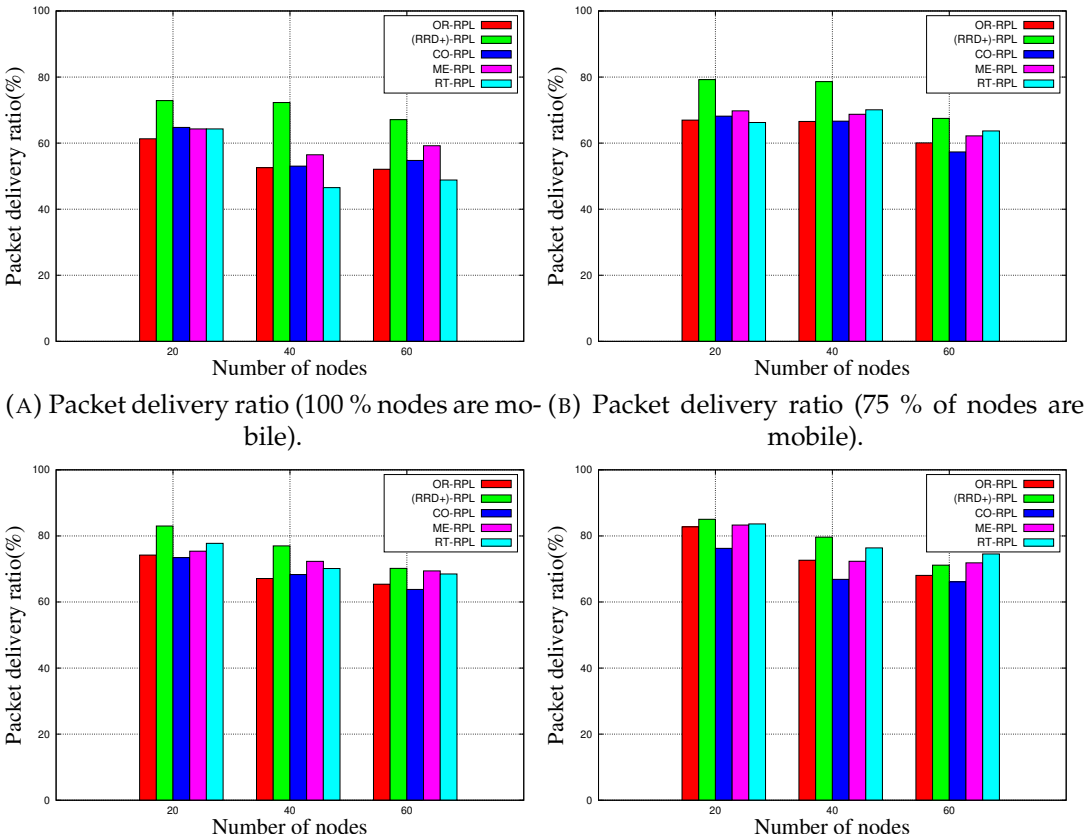
4.3.1 Packet delivery ratio

Figure 4.12 shows Packet delivery ratio of different RPL variants based on different degrees of mobility. RRD+ outperforms other standard RPL and its enhancement protocols when the degree of mobility is above 25 %. When the ratio of static nodes reaches 75 %, RRD+ no longer has an advantage and is even outperformed by MR-RPL and RT-RPL in 60-node scenarios as shown in figure 4.12d. This is mainly due to the fact that MR-RPL and RT-RPL have a mechanism to broadcast IDs of mobile nodes to other nodes in order to avoid being selected as next-hop nodes. Compared with mobile nodes, static nodes can offer more stable transmission paths. Therefore, MR-RPL and RT-RPL perform better when there are enough static nodes in the network to offer routes for the sink. On the other hand, MR-RPL and RT-RPL perform worse when there are not enough static nodes to chose from. Results also show that CO-RPL has limited contribution in mobility. This is mainly due to the fact that

TABLE 4.3: Simulations parameters used to evaluate RRD+ performance.

Item	**Parameters**
Network simulator	Cooja under contiki OS (3.0)
Radio propagation model	MRM with random behavior
Medium access control	CSMA/CA
Simulation time	5 minutes
Emulated platform	Z1 starter platform
Sensor Nodes Deployment	Random Deployment
Data size	30 Bytes
Transmission rate	1 pkt/sec
Transmission power	-20 dBm
Transmission range	[30 m, 50 m]
Number of nodes	20, 40, 60
Area of deployment	200 m * 200 m
Frequency range	2.4 GHz
Receive Sensitivity	-95 dBm
RRD+ RSSI $SAFE\_THRESHOLD$	-89 dBm
RRD+ RSSI $HYST\_THRESHOLD$	-92 dBm
RRD+ RSSI $HYSTERESIS$	-1 dB
Long lifetime $Long\_Lifetime$	30 s
Short lifetime $Short\_Lifetime$	15 s
$DEFAULT\_DIO\_INTERVAL\_MIN$	13
$DEFAULT\_DIO\_INTERVAL\_DOUBLINGS$	2
Number of iterations	10

CO-RPL does not offer a method to update C_ID (Corona ID of Corona mechanism) value.



(A) Packet delivery ratio (100 % nodes are mobile).

(B) Packet delivery ratio (75 % of nodes are mobile).

(C) Packet delivery ratio (50 % of nodes are mobile).

(D) Packet delivery ratio (25 % of nodes are mobile).

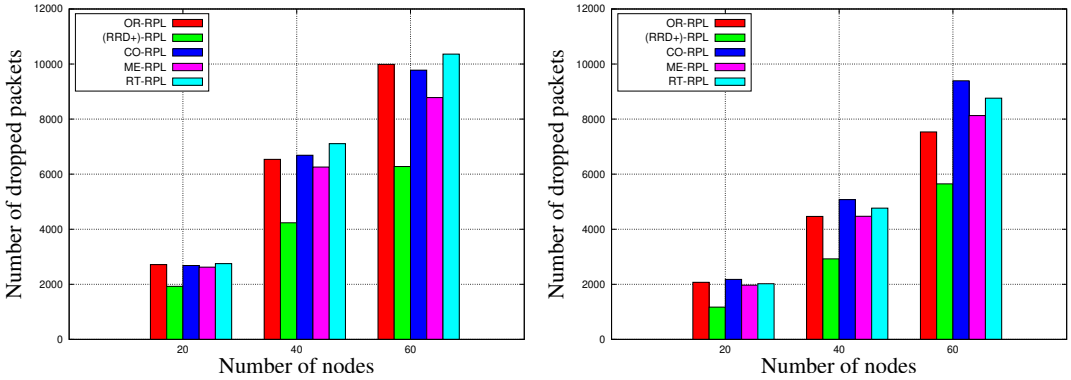
FIGURE 4.12: Packet delivery ratio.

4.3.2 Number of dropped packets

Figure 4.13 shows the number of dropped packets of different RPL variants based on different degrees of mobility. Results show that RRD+ outperforms all other RPL variants in all scenarios. This is mainly due to the fact that RRD+ helps RPL to adapt quickly to mobility and efficient Rank updating can pro-actively help nodes avoiding selecting a next-hop with a bad link quality. Therefore, RRD+ reduces retransmission attempts and the number of dropped packets is reduced as well. CO-RPL does not have proactive features, thus it suffers from a high number of dropped packets. With the increasing of the number of static nodes, ME-RPL and RT-RPL begin to show their performance for the reason discussed previously in packet delivery ratio results.

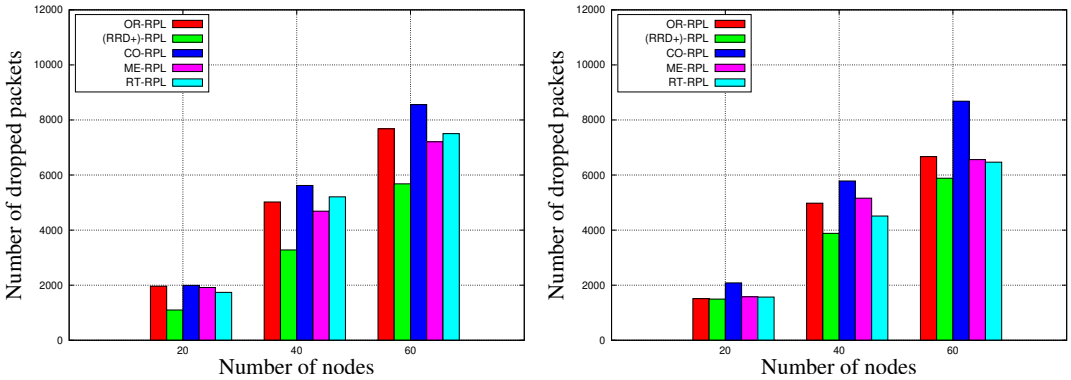
4.3.3 Average end-to-end delay

Figure 4.14 shows the average end-to-end delay of different RPL variants based on different degrees of mobility. The delay is computed on the delivered packets, lost packets do not appear in these results. Results show that RRD+ suffer from high average end-to-end delay compared to OR-RPL, ME-RPL and RT-RPL. The reason is that RRD+ cannot distinguish static nodes and mobile nodes. Therefore, compared



(A) Number of dropped packets (100 % nodes are mobile).

(B) Number of dropped packets (75 % of nodes are mobile).



(C) Number of dropped packets (50 % of nodes are mobile).

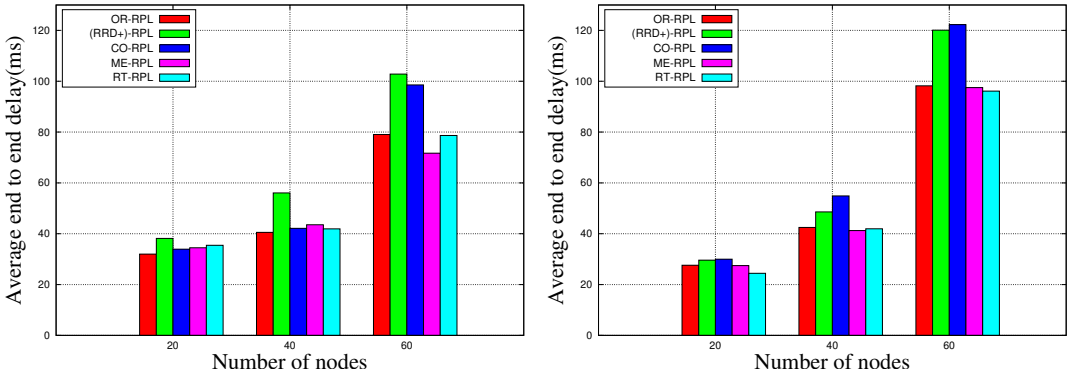
(D) Number of dropped packets (25 % of nodes are mobile).

FIGURE 4.13: Number of dropped packets.

with ME-RPL and RT-RP, RRD+ will select more mobile nodes as next-hops. Mobile nodes will indeed increase the delivery delay. CO-RPL also suffers from the same problem. The reason for CO-RPL is due to the fact that not updating C_ID would increase the number of hops during transmission, and hence, increase the delay. In addition, as we only compute delays for received packets, packets generated from lower Rank nodes will have a low average end-to-end delay compared to packets generated from higher Rank nodes. As with RRD+, higher rank nodes are able to successfully deliver their packets, this makes the average end-to-end delay of RRD+ higher.

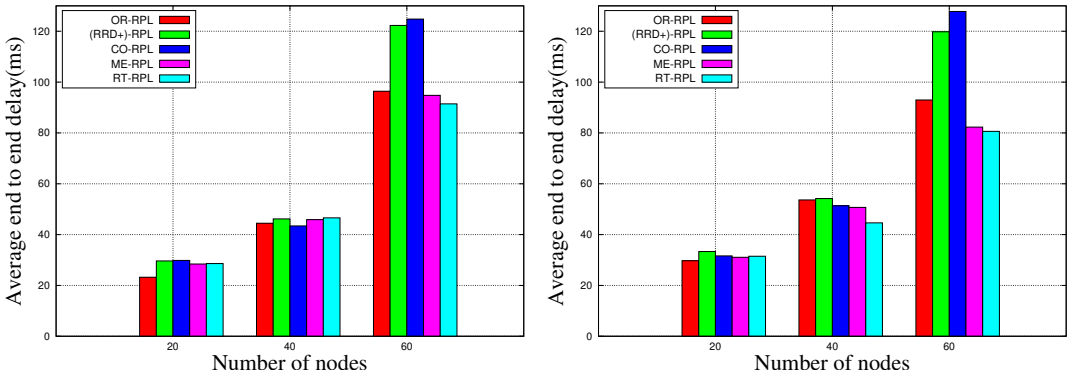
4.3.4 Number of control packets

Figure 4.15 shows the number of control packets of different RPL variants based on different degrees of mobility. Results show that RRD+ outperforms CO-RPL and ME-RPL in all scenarios. This is mainly due to the fact that ripple control message management helps reduce overhead according to Rank updating. Since RRD+ does not have a mechanism to distinguish static nodes and mobile nodes, there is no significant difference on overhead when the degree of mobility changes. CO-RPL periodically broadcasts DIO, therefore there is also no significant difference on overhead for different degrees of mobility. OR-RPL has the fewest overhead cost. This is mainly due to the fact that trickle algorithm, which is used in static networks, helps reduce overhead. ME-RPL suffers from a huge overhead cost when all nodes are mobile and gradually decreases the overhead when number of static nodes increases.



(A) Average end-to-end delay (100 % nodes are mobile).

(B) Average end-to-end delay (75 % of nodes are mobile).



(C) Average end-to-end delay (50 % of nodes are mobile).

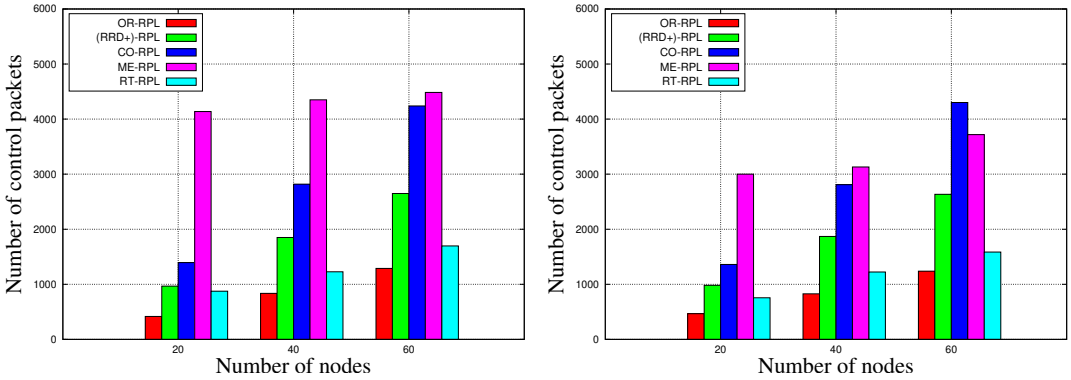
(D) Average end-to-end delay (25 % of nodes are mobile).

FIGURE 4.14: Average end-to-end delay.

This is mainly due to the fact that ME-RPL introduces a dynamic DIS management and ME-RPL will suffer from a huge number of DIS messages when there is more mobile nodes in the network. RT-RPL outperforms RRD+, CO-RPL and ME-RPL. The reason is that reverse trickle algorithm is a modified trickle algorithm, which can deal with mobility and reduces overhead at the same time. However, it cannot deal with random movement, thus compared with RRD+ it reduces overhead but does not increase packet delivery ratio.

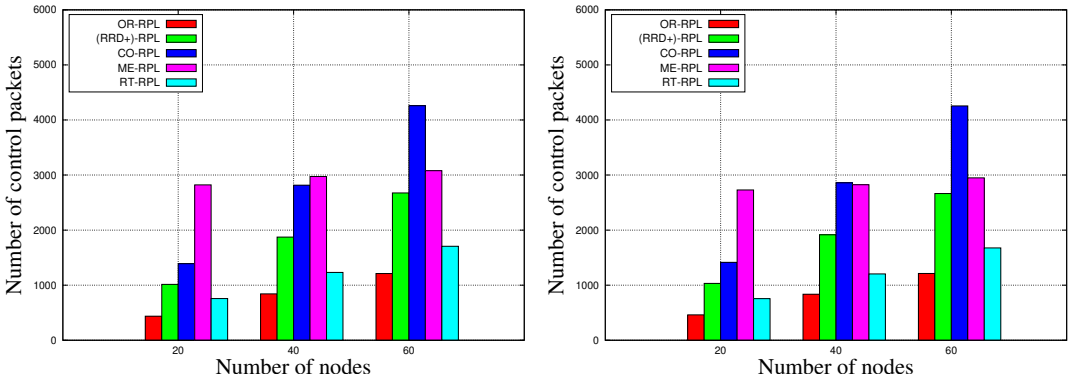
4.4 MRRD+ performance evaluation

We evaluated MRRD+ mechanism performance by comparing it to standard RPL using Cooja simulator. We tested MRRD+ and RPL in scenarios with 1, 2, 3 and 4 sinks. The deployment of sink(s) is shown in figure 4.16. We place the sinks in positions that allow a maximum coverage of the deployment area. Except the sinks, all other nodes are mobile and move within the $200m * 200m$ area. For RPL, we set *DEFAULT_DIO_INTERVAL_MIN* to 12 which results in *I_min* of 4 seconds and set *DEFAULT_DIO_INTERVAL_DOUBLINGS* value to 2 which results in *I_max* of 16 seconds. These configurations are decided after multiple simulation tests with different value combinations of *DEFAULT_DIO_INTERVAL_MIN* and *DEFAULT_DIO_INTERVAL_DOUBLINGS*. We found out with this configurations RPL would give attention to mobility detection in multi-sink scenarios and overhead saving meanwhile. Table 4.4 summarizes the rest of simulation settings.



(A) Number of control packets (100 % nodes are mobile).

(B) Number of control packets (75 % of nodes are mobile).



(C) Number of control packets (50 % of nodes are mobile).

(D) Number of control packets (25 % of nodes are mobile).

FIGURE 4.15: Number of control packets.

We retained six performance metrics to evaluate the efficiency of MRRD+ used on top of RPL compared to standard RPL: (i) packet delivery ratio, (ii) number of sent packets, (iii) number of retransmissions, (iv) number of dropped packets, (v) average end-to-end delay and (vi) number of control packets. For each network size, we generated 50 different mobility scenarios. Each performance metric is averaged over 50 iterations for each network size.

4.4.1 Packet delivery ratio

Figure 4.17 shows that with more sinks, both standard RPL and MRRD+ have better performance. This is mainly due to the fact that multiple sinks help reduce the number of hops, and thus reduces packet loss risk. From 20 nodes to 60 nodes MRRD+ improves packet delivery ratio by almost 15 % compared to standard RPL. This is mainly due to the fact that MRRD+ detects nodes movement and updates parents set in a timely manner, and then offers them next-hop selections with better link qualities.

4.4.2 Number of sent packets

Figure 4.18 shows that both MRRD+ and standard RPL have an improvement on the number of sent packets when the number of sinks increases. Multiple sinks in the topology reduces the number of hops for a packet to reach the sink and thus reduces the sending attempts. MRRD+ outperforms standard RPL in all scenarios.

TABLE 4.4: Simulations parameters used to evaluate MRRD+ performance.

Item	**Parameters**
Network simulator	Cooja under contiki OS (3.0)
Radio propagation model	MRM with random behavior
Medium access control	CSMA/CA
Simulation time	5 minutes
Emulated platform	Z1 starter platform
Sensor Nodes Deployment	Random Deployment
Data size	30 Bytes
Transmission rate	1 pkt/sec
Transmission power	-20 dBm
Transmission range	[30 m, 50 m]
Number of nodes	20, 40, 60
Area of deployment	200 m * 200 m
Frequency range	2.4 GHz
Receive Sensitivity	-95 dBm
MRRD+ RSSI $SAFE\_THRESHOLD$	-89 dBm
MRRD+ RSSI $HYST\_THRESHOLD$	-92 dBm
MRRD+ RSSI $HYSTERESIS$	-1 dB
Long lifetime $Long\_Lifetime$	30 s
Short lifetime $Short\_Lifetime$	15 s
$DEFAULT\_DIO\_INTERVAL\_MIN$	12
$DEFAULT\_DIO\_INTERVAL\_DOUBLINGS$	2
Number of iterations	50

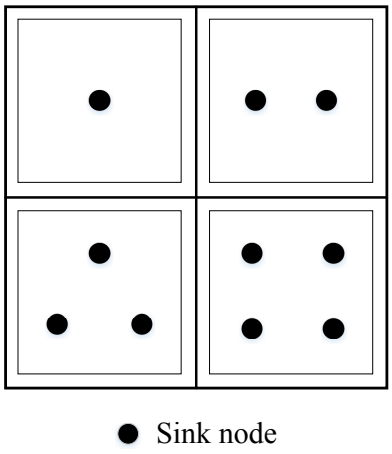


FIGURE 4.16: The positions of sink nodes in our 4 network scenarios.

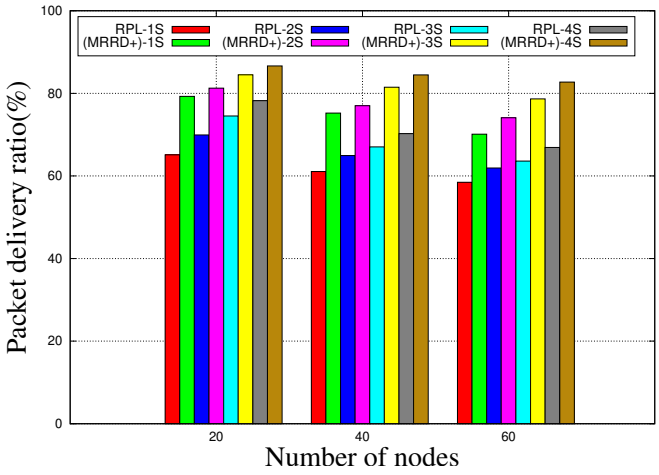


FIGURE 4.17: Packet delivery ratio.

Indeed, when the Rank value is updated timely, loops will be avoided. Thus, this reduces the number of hops travelled by data packets. The improvement for 40 and 60 nodes is even higher. Since with 40 and 60 nodes the Rank does not update in a timely manner without using MRRD+, more data packets will be transmitted through more hops, which causes more congestion and collisions.

4.4.3 Number of retransmissions

Figure 4.19 shows that MRRD+ outperforms RPL on different number of nodes especially with multiple sinks. Compared to RPL, MRRD+ has an obvious decrease in number of retransmissions when the number of sinks increases. This is mainly due to the fact that MRRD+ is able to update the Rank value in a timely manner and the updating of Rank value keeps the nodes in parents sets up-to-date. Proper next-hops selection would help reduce the retransmission attempts. Figure 4.19 also shows that MRRD+ has more retransmission attempts in one sink scenarios with 60 nodes. With 60 nodes, more nodes would update their next-hops in a timely manner and more packets would be travelling in the network which creates more congestion, and thus, more retransmissions. The problem only occurs for the scenario with one sink, where all the generated traffic would create high congestion around the sink node and the nodes near the sink.

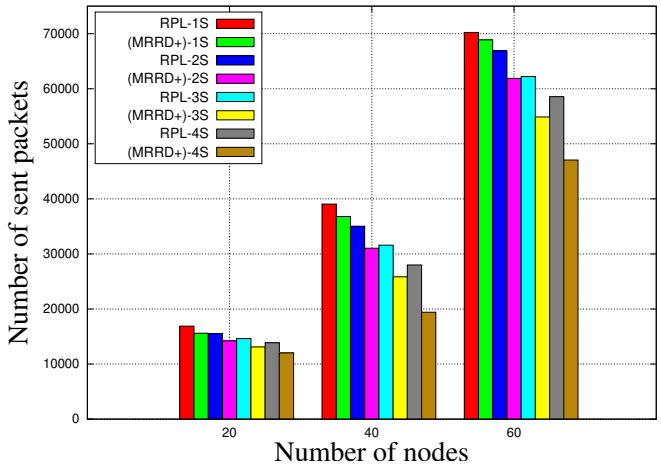


FIGURE 4.18: Number of sent packets.

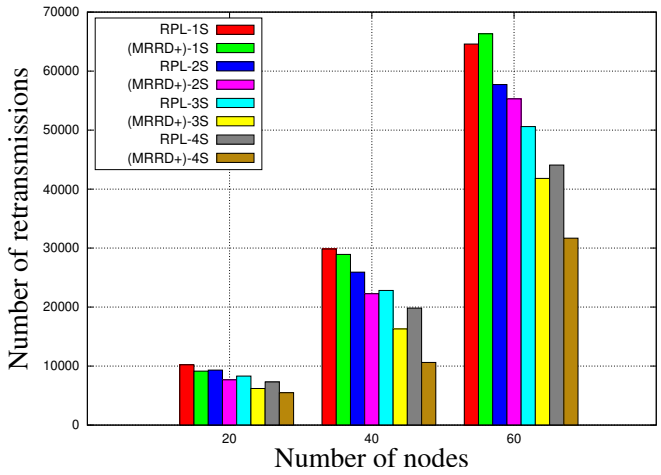


FIGURE 4.19: Number of retransmissions.

4.4.4 Number of dropped packets

Standard RPL suffers from a higher number of dropped packets in all the simulated scenarios compared to MRRD+. This is mainly due to the fact that MRRD+ is able to avoid using broken links for sending data packets to parent nodes that are out of reach are more quickly removed from the parents set.

4.4.5 Average end to end delay

Figure 4.21 shows that when the number of sinks increases, both MRRD+ and standard RPL reduce average end-to-end delay. This is mainly due to the fact that having more sinks helps reduce the average number of hops traveled by packets to reach the destination. Results also show that MRRD+ outperforms standard RPL especially in scenarios with multiple sinks mainly because Rank updating method in MRRD+ helps nodes detect movement and select a more suitable parent node faster. In addition, loops are avoided as we explained in subsection 4.4.2 when nodes move further away from the sinks which reduces the number of hops traveled by data packets. In such a way, data packets will reach the destinations much faster. Figure 4.21 also shows that MRRD+ has higher average end-to-end delay in one sink scenarios with

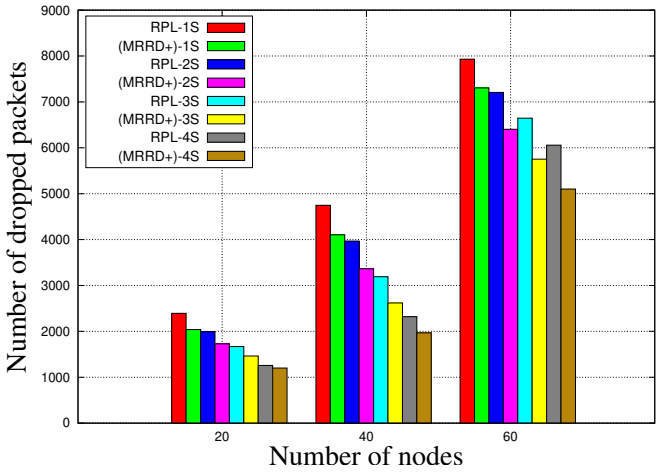


FIGURE 4.20: Number of dropped packets.

60 nodes. As we discussed for the number of retransmissions, in one sink scenarios with 60 nodes there would be more congestion in the network especially around the sink node which results in higher latency for MRRD+ because it is able to deliver more packet to sink and thus creates more congestion around the nodes that are closer to the sink. Compared to RPL, MRRD+ shows more retransmissions but fewer dropped packets, and this is the main reason for the higher delay with 60 nodes. The same as number of retransmissions, this problem only occurs for one sink scenarios with 60 nodes.

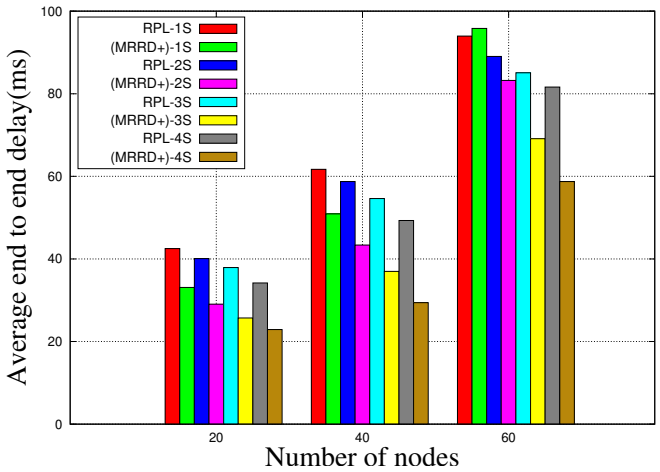


FIGURE 4.21: Average end-to-end delay.

4.4.6 Number of control packets

Figure 4.22 shows that the number of generated control traffic is not linear with regards to the number of nodes in mobility scenarios. Indeed, when a node moves it affects all its neighbors and thus these neighbors will generate control traffic. With more nodes in the network, the number of neighbors is higher and the number of control messages increases exponentially.

Figure 4.22 shows that MRRD+ have almost the same overhead cost compared to standard RPL when we set trickle algorithm *I_min* to 4 seconds and *I_max* to 16

seconds. Trickle algorithm is an efficient method originally designed for static networks. With the parameters we set, trickle algorithm can better cope with mobility and save overhead. MRRD+ dynamically manages the DIO interval according to movement direction and Rank values. According ripple control messages management, control messages can be broadcast timely but also dynamically reduced to a suitable frequency after an urgent sending. In such a way, MRRD+ has much better packet delivery ratio performance, but has almost the same overhead cost compared to standard RPL. This means our dynamic DIO interval management method is efficient in mobility and keeps the overhead within an acceptable cost. Results also show that even with more sinks, MRRD+ has similar results. Indeed, with lower Rank values on average, the DIO interval would be smaller according to our formula ($DIO\_interval = Base\_interval + Rank * Time\_unit$), but algorithm 2 helps maintain the same number of DIOs because Rank values change less often with more sinks.

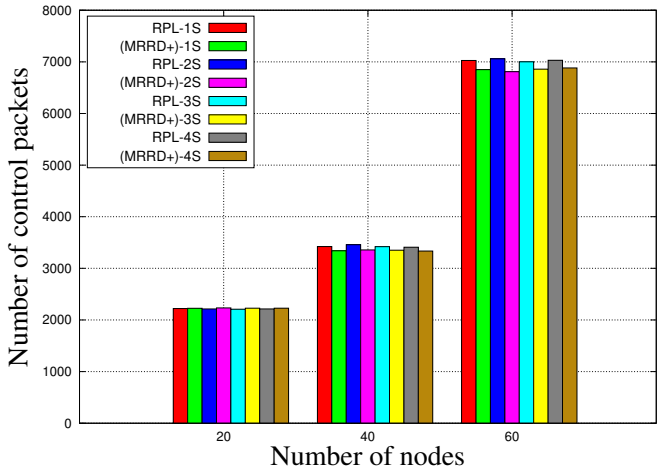


FIGURE 4.22: Number of control packets.

4.5 ADUP performance evaluation

We evaluated the performance of ADUP by doing simulations using Cooja simulator. At the beginning of the simulation, all nodes are randomly deployed within a $200mx200m$ area and they are free to move within this area. Every 5 seconds the velocity is randomly chosen from $[1m/s, 3m/s]$ and the direction of nodes changes by choosing a random destination position inside the deployment area. Under this mobility model, we set the minimum control message interval as 2 seconds according to the simulation tests, which means the minimum topology update interval is 2 seconds. In order to make sure that ID of next-hop updates can be sent to the sink timely, the upward transmission interval should be shorter than 2 seconds. The upward transmission rate we set is $1pkt/sec$. Every 1 second, all nodes except the sink generate a data packet to the sink. Meanwhile, the sink generates command packets to a randomly chosen node in the network every 1 second. Every scenario simulates 5 minutes of network activity. Table 4.5 summarizes the rest of simulation settings.

In order to assess the efficiency of ADUP in dealing with mobility, we compared it to other two existing protocols that cope with mobility: AODV and Flooding. In addition, AODV and Flooding are generic protocols that are designed for any application scenarios containing upstream and downstream traffic. AODV is a typical unicast-based routing protocol. It uses route discovery and route maintenance to

TABLE 4.5: Simulations parameters used to evaluate ADUP performance.

Item	Parameters
Network simulator	Cooja under contiki OS (3.0)
Radio propagation model	MRM with random behavior
Medium access control	CSMA/CA
Simulation time	5 minutes
Emulated platform	Sky starter platform
Sensor Nodes Deployment	Random Deployment
Data size	30 Bytes
Packet queue size	16
Upward transmission rate	1 pkt/sec
Downward transmission rate	1 pkt/sec
Transmission power	-20 dBm
Transmission range	[30 m, 50 m]
Number of nodes	20, 40, 60
Area of deployment	200 m x 200 m
Frequency range	2.4 GHz
Reception threshold	-95 dBm
Number of iterations	10

support dynamic topologies. Furthermore, it is based on periodic advertisements and distance vector routing, which is more adaptive to mobile scenarios compared to DSR for example. Flooding is a broadcast-based routing protocol which is not sensitive to mobility. It may support mobile scenarios well with few transmission events. It is also interesting to estimate the performance of Flooding in mobile scenarios with different traffic patterns. In order to limit the number of transmissions using Flooding routing protocol, nodes only route the same packet once. We used unique identifiers for packets in order to manage this issue.

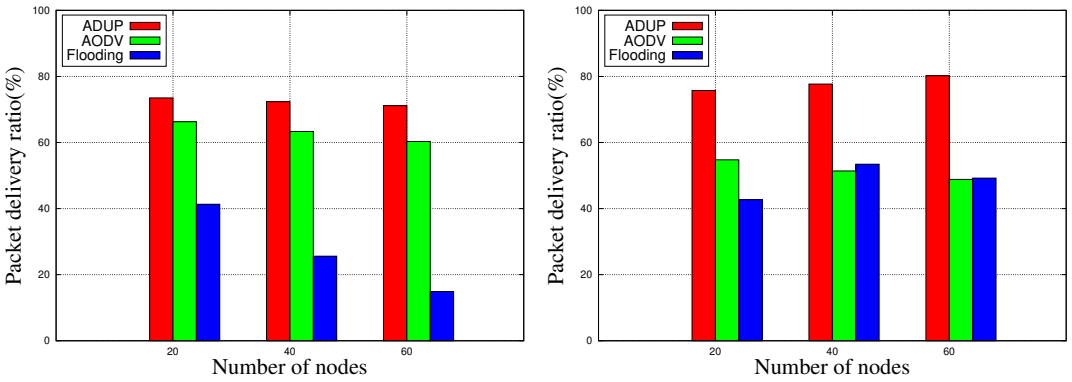
We used four performance metrics to evaluate the efficiency of these protocols: (i) packet delivery ratio, (ii) average end-to-end delay, (iii) dropped packet ratio and (iv) number of control packets. For each network size, we generated 10 different random mobility scenarios. Each performance metric is averaged over 10 iterations for each network size.

4.5.1 Packet delivery ratio

Figure 4.23a shows that ADUP outperforms AODV and Flooding in terms of packet delivery ratio for upstream traffic. ADUP has about 10 % improvement over AODV. This is mainly due to the fact that in upward routing ADUP only needs to update next-hops rather than whole path routes, which helps adapt faster to mobility. Moreover, ADUP and AODV are much better than Flooding. The reason is that Flooding uses broadcasting as a basic traffic pattern which causes serious network congestion especially when the number of senders increases.

Figure 4.23b shows that ADUP is much better than AODV and Flooding for downstream traffic. This is mainly due to the fact that benefiting from upstream traffic ADUP allows the sink to update the path to reach any node in the network in a timely manner. Results also show that packet deliver ratio of AODV decreases when the number of nodes increases. The reason is that with more nodes upstream

traffic will be higher which causes more congestion and risk of collisions. Affected by upstream traffic, the downstream traffic of AODV would lose more packets during transmission. However, unlike AODV, the packet delivery ratio of Flooding on downstream traffic increases for 40 nodes first and then decreases for 60 nodes. This is mainly due to the fact that Flooding uses broadcasting rather than unicasting. When the number of nodes increases, broadcasting would help the same packet being relayed more times, which would increase the opportunity of successfully sending a packet to reach the destination. This allows Flooding to increase its packet deliver ratio of downstream traffic with 40 nodes. However, the downstream traffic of Flooding is also affected by the upstream traffic. When there is serious network congestion caused by upstream traffic of 60 nodes, the packet delivery ratio of downstream traffic decreases.



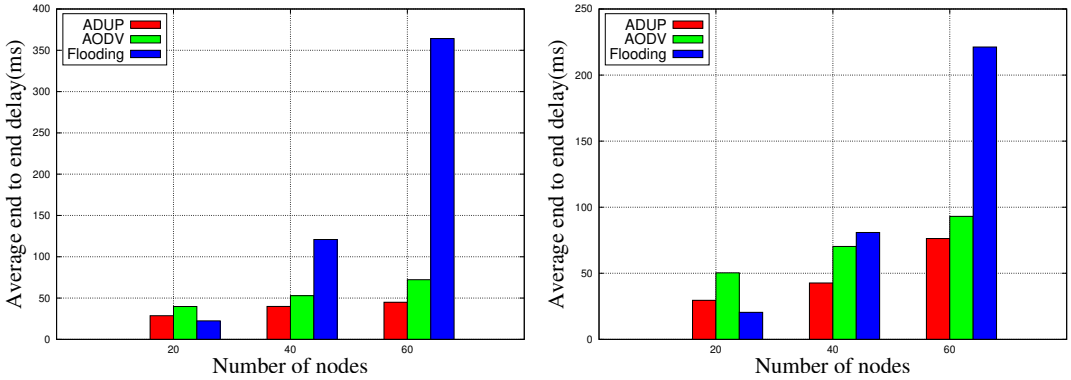
(A) Packet delivery ratio (upstream traffic). (B) Packet delivery ratio (downstream traffic).

FIGURE 4.23: Packet delivery ratio.

4.5.2 Average end-to-end delay

Figure 4.24a shows that ADUP outperforms AODV on average end-to-end delay for upstream traffic. This is mainly due to the fact that ADUP can quickly adapt to mobility and dynamically update parents set according to the movement of nodes, which reduce time interval before sending a packet. However, AODV needs to build routes before transmission. This process will be more frequent in mobility, since link failure happens more often, which results in longer delays before sending a packet with AODV. Compared with ADUP and AODV, the average end-to-end delay of Flooding increases exponentially when the number of nodes increases. The reason is that Flooding broadcasts packets to the entire network which causes network congestion. Congestion causes packets to be delayed at each hop until they reach the sink. It is worth noticing that although ADUP and AODV achieve similar average end-to-end, ADUP was able to reach that delay for a higher packet delivery ratio.

Figure 4.24b shows that the average end-to-end delay of ADUP outperforms that of AODV and Flooding for downstream traffic. This is mainly due to the fact that upstream traffic helps ADUP to quickly update route information. Packets sent by the sink are relayed to the destination in a timely manner. The same as upstream traffic, the average end-to-end delay of Flooding on downstream increases as well. Network congestion caused by upstream traffic is also the reason.



(A) Average end-to-end delay (upstream traffic).

(B) Average end-to-end delay (downstream traffic).

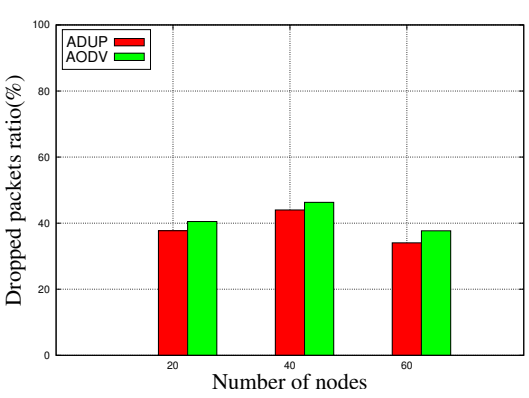
FIGURE 4.24: Average end-to-end delay.

4.5.3 Dropped packets ratio

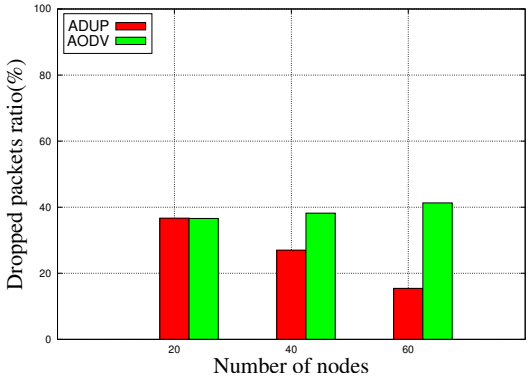
Figure 4.25 shows dropped packets ratio for different traffic patterns. Due to the fact that Flooding directly broadcasts packets, there are no retransmissions, and thus no dropped packets. Hence, in figure 4.25 we only show comparison between ADUP and AODV.

Figure 4.25a shows that ADUP outperforms AODV in terms of dropped packets ratio for upstream traffic. This is mainly due to the fact that link quality monitoring and rank updating helps ADUP detect movement quickly and select next-hop nodes with good link quality. This increases the success rate of sending a packet and reduces the number of dropped packets. Results also show that dropped packets ratio of ADUP and AODV for upstream traffic increases with 40 nodes first and then decreases with 60 nodes. When the number of nodes increases from 20 to 40, there would be more upstream traffic, which would cause more collisions and retransmissions. Thus, the dropped packets ratio increases with 40 nodes. However, when the number of nodes continues to increase, the network density becomes higher. A node would have more available next-hops to select from. This helps nodes to select next-hops with better link quality. This results in fewer number of dropped packets.

Figure 4.25b shows that ADUP also outperforms AODV in terms of dropped packets for downstream traffic and the ratio of ADUP even decreases when the number of nodes increases. Benefiting from upstream traffic and route building in downward routing, ADUP helps the sink node to build downward routes to reach any node in the network. When the number of nodes increases, the density of nodes increases. Next-hop nodes with better link quality would be used by the sink and this would enhance the path quality between the sink and the destinations. This also helps reducing the number of dropped packets. However, unlike ADUP, the dropped packets ratio of AODV for downstream traffic increases slightly when the number of nodes increases. This is mainly due to the fact that AODV is a reactive protocol and it builds routes only when there is data to be transmitted. This process delays transmission, which results in AODV not being able to update topology in a timely manner. This would cause packet loss due to inappropriate next-hop selection.



(A) Dropped packet ratio (upstream traffic).



(B) Dropped packet ratio (downstream traffic).

FIGURE 4.25: Dropped packet ratio.

4.5.4 Number of control packets

The number of control packets is the sum of control packets that are sent during the simulation by the routing protocol. ADUP only contains one type of control message which is broadcast by the sink node and propagated by other nodes until it reaches the leaf nodes. AODV generates three types of control messages: Route Request (RREQ) messages, Route Reply (RREP) messages and Route Error (RERR) messages. RREQ is used in route discovery process in order to build a route to reach the destination node. RREP is sent once the destination node receives a RREQ or an intermediate node has an active route to the destination. RERR is sent whenever a node detects a link failure or does not have an active route to the destination. Flooding only uses broadcasting as traffic patten and it does not need any additional control packets. Thus, in figure 4.26 we only show comparison between ADUP and AODV without considering the length of control packets.

Figure 4.26 shows that ADUP has very low overhead compared to AODV, especially when the number of nodes increases. This comes from two reasons. The first reason is that ADUP uses dynamic control message management to reduce the number of control packets used by upstream traffic, which helps to save overhead for upstream traffic. The second reason is that ADUP embeds the information of next-hop in data packets destined to the sink. This increases the load of data packets but allows the sink to build routes for downstream traffic without generating additional control traffic. Moreover AODV broadcasts RREQ to ask for a route, which adds more overhead to the network especially when the number of nodes increases.

4.6 Experimental evaluation

Evaluating the routing protocol by experiment helps know its behavior and efficiency in the physical work. In this section we present the experimental results of our contribution ADUP using sensor nodes. Firstly, we describe the experiment environment, present the testing scenarios and introduce the network set-up. Secondly, we evaluate ADUP and RPL by packet deliver ratio and number of dropped packets from both experimentations and simulations.

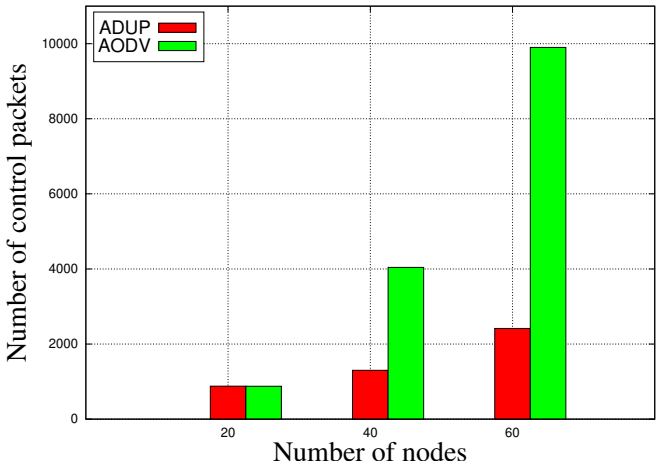


FIGURE 4.26: Number of control packets.

4.6.1 Experiment environment and network set-up

Our experiment is conducted indoors in a practical building. Multiple Wi-Fi APs were active on the 2.4 GHz band in the building. The experiment is done using TelosB motes communicating on the channel number 26 of IEEE 802.15.4 standard operating on the unlicensed 2.4 GHz band. This channel is the least exposed channel to interference from Wi-Fi.

We set up nodes in corridor and a room as shown in figure 4.27 (scenario 1) and figure 4.28 (scenario 2). Nodes are placed on the ground or tables. The transmission power is set at the minimum level -25 dBm to ensure a multi-hop topology network. We design two different scenarios to evaluate our work.

In scenario 1 shown in figure 4.27, we expect to test ADUP in dealing with process that a mobile node changes parent nodes with different Rank values during movement. We deploy 8 static nodes along the two sides of corridor. The distance between two nearby nodes in one side is 15 meters, which is a suitable distance that offers stable link between two nodes after testing. One person with a sensor node on his body moves with a random velocity ranging from $1m/s$ to $3m/s$ following the trajectory of dashed line in figure 4.27. One sink node is placed at the end of corridor to collect information. When data packet is received at the sink, its number of hops from the source node is printed. We notice that the topology is stable and the number of hops from static nodes to the sink keeps the same during testing. The two nodes that are nearest to the sink are both 1 hop away. The two farthest nodes from the sink are both 4 hops away. In the network, the 8 static nodes and the mobile node send 1 packet to the sink per second. Meanwhile the sink sends 1 packet, 5 packets or 10 packets per second to the mobile node.

In the scenario 2 shown in figure 4.28, we expect to test the performance of ADUP in a more compact space with more mobile nodes. We deploy 12 static nodes in the confined space. 9 static nodes are deployed along the two sides of corridor. The distance between two nearby nodes in one side is 10 meters. Compared with 15 meters set for scenario 1, 10 meters offers more stable link. However, collisions are easier happened due to close distance between two nodes. Under this setting we could test the perfomance of ADUP in compact space. 3 static nodes are deployed in a room. One sink node is placed at the corner of the corridor. 3 persons with one sensor node on their body randomly move within the space. The velocity randomly ranges from $1m/s$ to $3m/s$. Unlike the scenario 1, static nodes in the scenario 2 do not keep the

same hops during testing, since the density of node is higher. The maximum number of hops in this scenario changes from 4 to 5. In this network, 12 static nodes and 3 mobile nodes send 1 packet to the sink per second. Meanwhile the sink randomly selects one mobile node from the three and sends to it by a transmission rate of 1 pkt/sec, 5 pkt/sec or 10 pkt/sec.

We keep the simulation parameter *SAFE_THRESHOLD* as -89 dBm, *HYST_THRESHOLD* as -92 dBm and *HYSTERESIS* as -1 dB. For RPL, we set *DEFAULT_DIO_INTERVAL_MIN* to 11 which results in *I_min* of 2 seconds and set *DEFAULT_DIO_INTERVAL_DOUBLINGS* value to 3 which results in *I_max* of 16 seconds. We set RPL in the storing mode, which is more efficient in downstream traffic compared with non-storing mode. We used two performance metrics to evaluate the efficiency of ADUP compared to standard RPL: (i) packet delivery ratio and (ii) number of dropped packets. For each scenario, we generated 10 different mobility scenarios. Each performance metric is averaged over 10 iterations for each network size. Table 4.6 summarizes the rest of experiment settings.

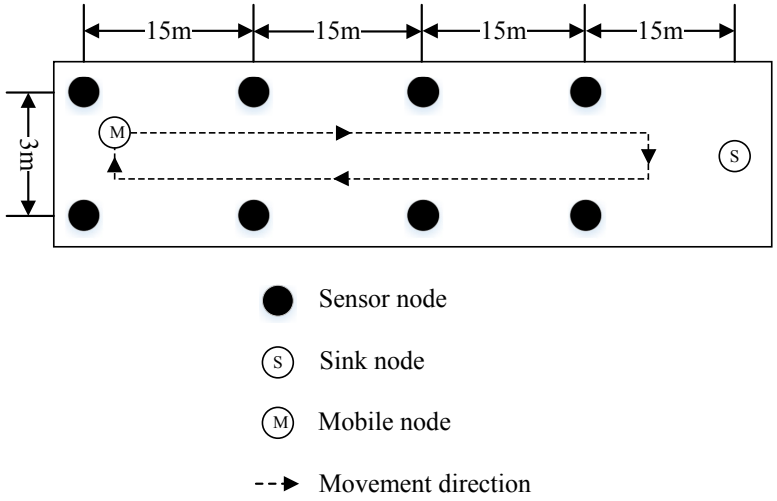


FIGURE 4.27: Testing scenario 1.

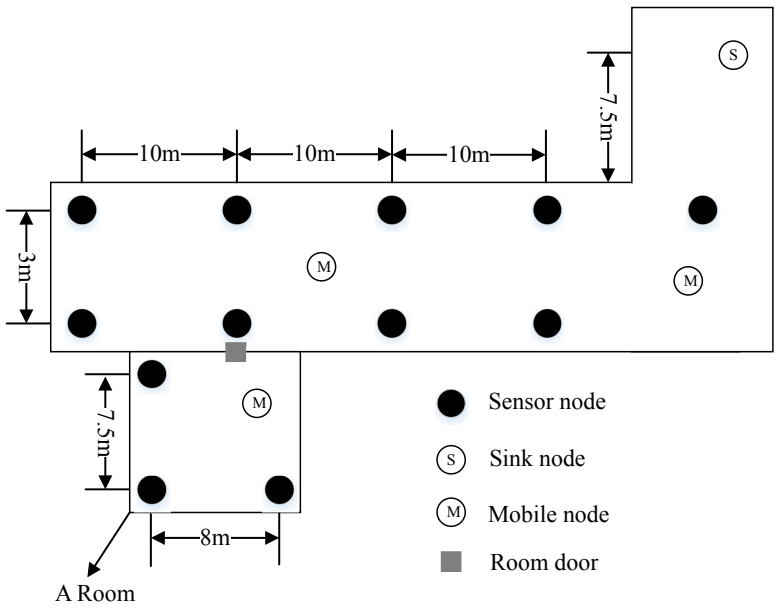


FIGURE 4.28: Testing scenario 2.

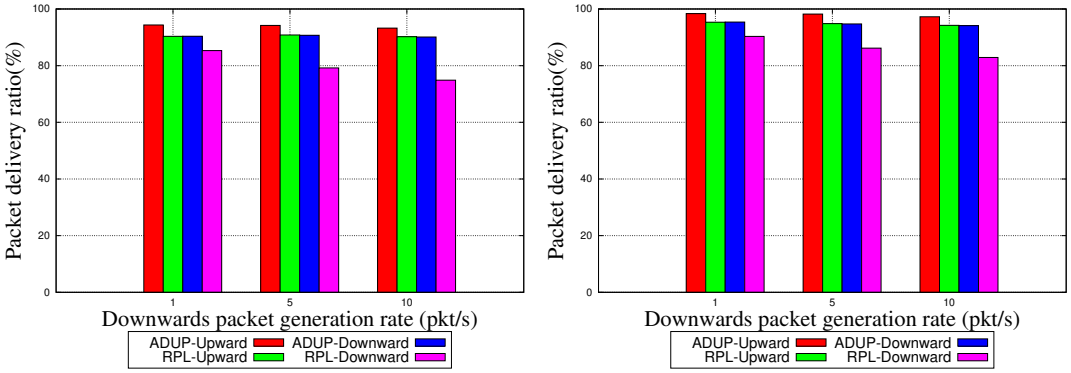
TABLE 4.6: Experiment parameters used to evaluate ADUP performance.

Item	Parameters
Medium access control	CSMA/CA
Simulation time	5 minutes
Platform	TelosB platform
Data size	30 Bytes
Number of motes	10 and 16
Number of mobile motes	1 and 3
Upward transmission rate	1 pkt/sec
Downward transmission rate	1,5,10 pkt/sec
Transmission power	-25 dBm
Receive Sensitivity	-90 dBm(min) and -95 dBm(max)
RRD+ RSSI *SAFE_THRESHOLD*	-89 dBm
RRD+ RSSI *HYST_THRESHOLD*	-92 dBm
RRD+ RSSI *HYSTERESIS*	-1 dB
Frequency range	2.4 GHz
Number of iterations	10
Mode of RPL	Storing mode

4.6.2 Packet delivery ratio for experimental scenarios

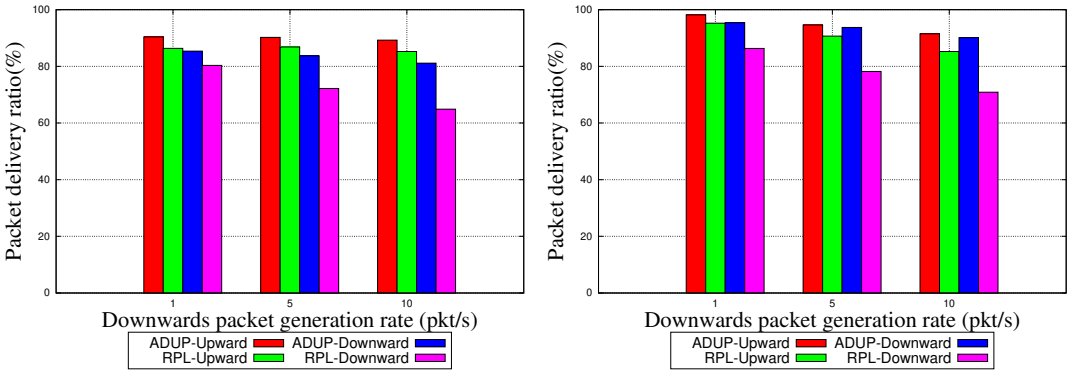
Figure 4.29 presents the packet delivery ratio results of scenario 1 for both experiment and simulation. The figure 4.29a presents the experiment results and the figure 4.29b shows the simulation results. We notice that the simulation results outperform experiment results. Indeed, in physical world, the external environment has a negative impact on the radio signal. Several Wi-Fi operating on the same frequency band 2.4 GHz are deployed around the experiment area, which increases the probability of interference. In addition, the link qualities in the experiment are not the same as those of the simulated topology. The results show that ADUP and RPL have similar packet delivery ration on upstream traffic. This is mainly due to the fact that there is only one mobile node in the network and most upstream traffic comes from static nodes. The enhancement of ADUP for mobility is not obvious with few number of mobile nodes. The results also show that ADUP outperform RPL on downstream traffic and there is significant enhancement on packet deliver ratio when the downwards packet generation rate increases. This is mainly due to the fact that benefiting from upstream traffic ADUP allows the sink to update the path to reach any nodes in the network in a timely manner.

Figure 4.30 presents the packet delivery ratio results of scenario 2 for both experiment and simulation. The figure 4.30a presents the experiment results and the figure 4.30b shows the simulation results. The same as packet delivery ratio results of scenario 1, simulation results outperform experiment results, which comes from the same reason discussed above. We notice that ADUP has obvious enhancement on upstream traffic comparing with RPL. Indeed, there are more mobile nodes in scenario 2. ADUP could help them cope with mobility and select the next-hop to reach the sink. With the increasing of downwards packet generation rate, ADUP shows its efficiency on downstream traffic compared with RPL. Indeed, ADUP could quickly help the sink build route path to reach the three mobile nodes concurrently.



(A) Packet delivery ratio (experiment). (B) Packet delivery ratio (simulation).

FIGURE 4.29: Packet delivery ratio of scenario 1.



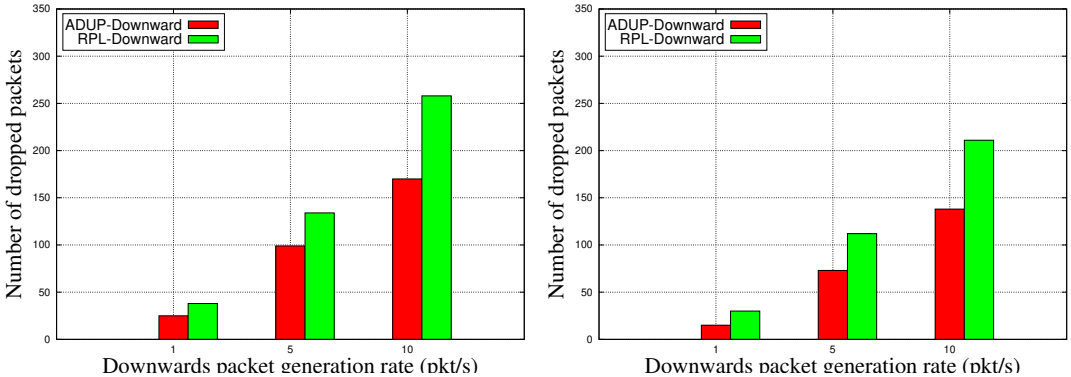
(A) Packet delivery ratio (experiment). (B) Packet delivery ratio (simulation).

FIGURE 4.30: Packet delivery ratio of scenario 2.

4.6.3 Number of dropped packets for experimental scenarios

Due to the limit of equipment, we do not have enough monitoring devices connected to each sensor to collect transmission information. We only connect monitoring devices to the sink node and the three mobile nodes. Thus, we can only show the number of dropped packets from downstream traffic. Figure 4.31 shows that the number of dropped packets from downstream traffic of scenario 1 for both experiment and simulation. The figure 4.31a presents the experiment results and the figure 4.31b shows the simulation results. The results show that ADUP outperforms RPL in terms of dropped packets. This is mainly due to the fact that ADUP helps the sink node build downward routes to reach the mobile node in a timely manner, which would reduce the number retransmissions due to lossy link. And it will future reduce the number of dropped packets. Indeed, with mobility, parent nodes will change frequently. Compared with ADUP, it is hard for RPL to send DAOs timely to keep the downward routing table up-to-date and packets will be dropped due to outdated path information.

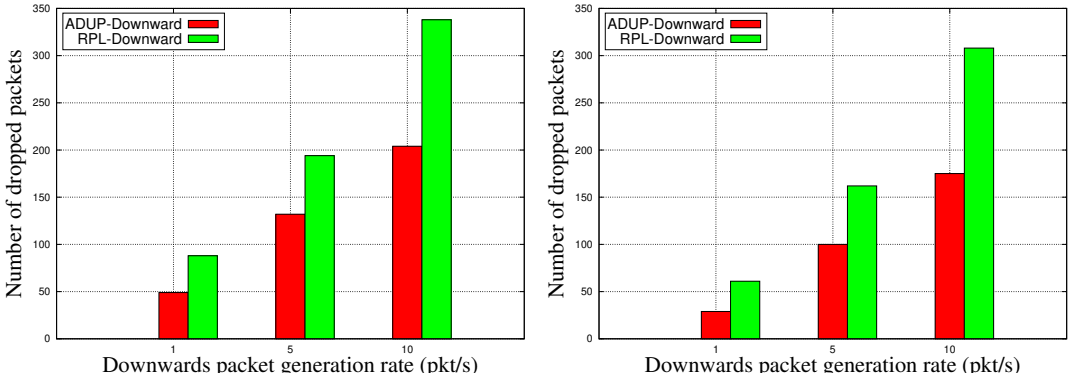
Figure 4.32 shows that the number of dropped packets from downstream traffic of scenario 2 for both experiment and simulation. We noticed that in scenario 2 the sink drops more packets compared with scenario 1. Indeed, with more mobile nodes, it would be more difficult for the sink to build path routes to all the mobile nodes concurrently. However, unlike RPL, ADUP does not have much increase on the number of dropped packet in scenario 2. This is mainly due to the fact that



(A) Number of dropped packets (experiment). (B) Number of dropped packets (simulation).

FIGURE 4.31: Number of dropped packets of scenario 1.

benefiting from upstream traffic ADUP allows the sink easily build the path to all the mobile nodes simultaneously.



(A) Number of dropped packets (experiment). (B) Number of dropped packets (simulation).

FIGURE 4.32: Number of dropped packets of scenario 2.

4.7 Summary

In this chapter, we present the simulation and experiment results for ADUP.

We evaluate RL mechanism while using different link metrics. Simulation results show that our RL mechanism enhances the network performance on many levels. We have proven that random selection of parent nodes in parents set has a good effect on the network performance because it avoids sending packets to the same node even though it is moving away. We have also proven that using slow convergence routing metric such as Latency metric is not suitable for mobile scenarios.

We applied RL mechanism in RPL and proved that RL mechanism enhances RPL performance but suffers from high overhead. RRD is proposed to enhance overhead performance of RL. The results show that RRD successfully helps nodes send more packets to the sink with less retransmissions and overhead.

Based on RRD, RRD+ is proposed to enhance the performance of nodes that are close to the edge of transmission. We compared RRD+ to other existing mobility enhancements for RPL and standard RPL. Simulation results show that RRD+ mechanism enhances RPL on many levels: successfully delivery packets, packet loss,

number of dropped packets, end-to-end delay and overhead. The results also show that RRD+ is suited for networks where more than 25 % of nodes are mobile nodes. When there are less mobile nodes in the network, RRD+ has limited contribution on network performance.

We compared MRRD+ to standard RPL in multi-sink mobile scenarios. We tested them in scenarios with 1, 2, 3 and 4 sinks. Simulation results show that our MRRD+ mechanism enhances RPL with multiple sink. Results also show that the efficiency of MRRD+ increases when the number of sinks in the network increases.

We compared ADUP with two other generic routing protocols AODV and Flooding. Results show that ADUP outperforms AODV and Flooding on different performance metrics in mobility scenarios. Results also show that ADUP supportd upward and downward routing simultaneously in dense and highly mobile scenarios. In the upward routing process, ADUP helps nodes to detect next-hops with good link quality. In the downward routing process, ADUP helps the sink node to build routes to reach any node in the network (given that this node is generating traffic towards the sink).

In order to know the behavior and efficiency of our work in the physical work we compared the performance of ADUP and standard RPL by experiment. The experimental shows that ADUP outperforms standard RPL on packet delivery ration and number of dropped packets.

Obtained results in both simulation and experiment qualify the contribution of each of our protocol variants.

Chapter 5

Conclusion and perspectives

In this chapter we conclude the thesis by reminding the addressed problem, summarizing our contributions, and highlighting some perspectives.

5.1 Conclusion

Dealing with mobility in WSNs and LLNs is a challenging task for a compromise between efficiency and complexity. One of the most dominant routing protocols designed for LLNs, RPL, was original specified without any special support for mobility. This is the major drawback that prevents many applications with mobility from using it. The goal of this thesis is to propose an efficient mobility support for routing protocols (like RPL) in LLNs. We focus on convergecast and anycast, which are the most used traffic modes in LLNs, in mobile network scenarios.

Firstly we proposed a series of protocols (RL, RRD, RRD+ and MRRD+) to support mobility in convergecast LLNs. These protocols can easily be applied to RPL routing protocol. RRD+ and MRRD+ are enhanced versions of RL and RRD, and MRRD+ is a variant of RRD+ to support multiple sinks in convergecast LLNs. In these protocols we monitor RSSI values and updates Rank values accordingly in order to avoid loops, and they also dynamically manages the interval of control messages. We monitor the movement of moving nodes based on the variation of RSSI values. When a node detects that one of its potential next-hops in the parents set is moving away, it will anticipate a link failure and try to use another node from the parents set. This helps nodes update their next-hop choice in a timely manner in mobile scenarios.

In succession we proposed ADUP, a routing protocol that supports convergecast and anycast concurrently in mobile scenarios. It is suitable for application where all nodes are mobile and send periodical data packets to a sink node. In addition, this sink needs to periodically contact other nodes of the network. The support of convergecast in mobility is based on RRD+. The support of anycast in mobility is an extension work on RRD+. Nodes embed the best next-hop of nodes in the packet headers and send it to the sink inside the periodical data packets. Once the sink receives the data packets, it will build a next-hop table. Based on this table, the sink is able to build the route to reach any node in the network. Due to the fact that the best next-hop is selected from a dynamic parents set periodically, the next-hop table is also dynamic and could adapt to mobility in a timely manner.

We implemented our work in Cooja and compared them with other routing protocols. Simulation results show the efficiency of our work on many levels. We also evaluated ADUP by experiment and compared it to standard RPL. The obtained results show the ability of ADUP to help mobile nodes increase packet deliver ratio and reduce number of dropped packets. The experiment is a proof of concept of our

contributions. However, the number of mobile nodes used in the network scenarios does not put enough emphasis on the addressed problem but these first results obtained with real sensor nodes remain positive.

5.2 Perspectives

The contributions of this thesis can be extended in several direction. In what follows we enumerate some of them.

5.2.1 Energy consumption optimization

Energy consumption is a critical problem we need to investigate in the future. Right now we do not use sleep mode in our work, since topology information updating relies on information broadcasting and collection in a timely manner and a sleep mode would delay this process. However, without sleep mode nodes are easily in the condition of energy depletion. Thus we plan to integrate a dynamic duty cycle in RRD+ according to different degrees of mobility in order to reduce energy consumption.

5.2.2 Optimizing length of packet headers

In order to support downstream traffic in ADUP, we put the route path information in the packets header. However, with the constrained wireless layers IEEE 802.15.4, the maximum frame size is 127 bytes (including header). The huge size of packet header would increase network capacities and power consumption. In order to support large scale networks with hundreds of nodes, it is necessary to introduce a mechanism in order to encode downward routes in ADUP, which enables a reduction of the length of packet headers.

5.2.3 Different degrees of mobility support

The simulation results show that RRD+ is suited for the networks where more than 25% of nodes are mobile. Indeed, RRD+ is proposed to support highly mobile scenarios. However, in many applications, only some of the network nodes need to be mobile. In order to cope with the scenarios with only few number of mobile nodes, we plan to add a mechanism for RRD+ to distinguish static nodes from mobile nodes in order to improve the performance of RRD+ in different degrees of mobility.

5.2.4 Fault tolerance

Compared with static networks, node failures easily happen in mobile networks. Most of the time, node failure happens unexpectedly. Our work is unable to detect that the certain node that is in a fault and then finding a solution to avoid sending traffic to this node. Moreover, our work cannot decide whether the node is out reach or in a fault. Thus, in the future we plan to integrate a fault tolerant method [58] with ADUP to help nodes in the network distinguish whether a node is beyond reachable of in a fault and then find alternative routes to reach the final destination.

Publication

International journal papers

- **Jinpeng Wang**, Gérard Chalhoub, Michel Misson. Adaptive Downward/Upward Routing Protocol for Mobile-Sensor Networks. Future Internet. 2019 Jan; 11(1):18.
- **Jinpeng Wang**, Gérard Chalhoub. Mobility support enhancement for RPL with multiple sinks. Annals of Telecommunications. 2019 Jan 2:1-4.
- **Jinpeng Wang**, Gérard Chalhoub. Study of mobility enhancements for RPL in convergecast scenarios. Future Internet. 2017 Dec; 9(4):86.

International conference papers

- **Jinpeng Wang**, Gérard Chalhoub, Michel Misson. Mobility support enhancement for RPL. In 2017 International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN) 2017 Nov 28 (pp. 1-6). IEEE. **(Best paper in conference)**
- Gérard Chalhoub, Hamadoun Tall, **Jinpeng Wang**, Michel Misson. DFTR: Dynamic Fault-Tolerant Routing Protocol for Convergecast WSNs. In 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall) 2017 Sep 24 (pp. 1-7). IEEE.
- **Jinpeng Wang**, Gérard Chalhoub, Hamadoun Tall, Michel Misson. Routing protocol enhancement for mobility support in wireless sensor networks. In International Conference on Ad-Hoc Networks and Wireless 2017 Sep 20 (pp. 262-275). Springer, Cham.

International competition abstract

- **Jinpeng Wang**, Hamadoun Tall, Gérard Chalhoub. Competition: Smart flooding with multichannel for industrial wireless sensor networks. In Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks 2018 Feb 16 (pp. 219-220). Junction Publishing.

Bibliography

1. Rashid, B. & Rehmani, M. H. Applications of wireless sensor networks for urban areas: A survey. *Journal of network and computer applications* **60,** 192–219 (2016).
2. Erman-Tüysüz, A. *Multi-sink mobile wireless sensor networks: dissemination protocols, design and evaluation* PhD thesis (Universiteit Twente, 2011).
3. Rawat, P., Singh, K. D., Chaouchi, H. & Bonnin, J. M. Wireless sensor networks: a survey on recent developments and potential synergies. *The Journal of supercomputing* **68,** 1–48 (2014).
4. Aminian, M. & Naji, H. R. A hospital healthcare monitoring system using wireless sensor networks. *J. Health Med. Inform* **4,** 121 (2013).
5. Kamgueu, P. O., Nataf, E. & Ndie, T. D. Survey on RPL enhancements: A focus on topology, security and mobility. *Computer Communications* **120,** 10–21 (2018).
6. Oliveira, A. & Vazão, T. Low-power and lossy networks under mobility: A survey. *Computer Networks* **107,** 339–352 (2016).
7. Perkins, C., Belding-Royer, E. & Das, S. *Ad hoc on-demand distance vector (AODV) routing* tech. rep. (IETF, 2003).
8. Chakeres, I. D. & Klein-Berndt, L. AODVjr, AODV simplified. *ACM SIGMOBILE Mobile Computing and Communications Review* **6,** 100–101 (2002).
9. Montenegro, G. & Kushalnagar, N. *AODV for IEEE 802.15. 4 networks* tech. rep. (IETF, 2005).
10. Gomez, C., Salvatella, P., Alonso, O. & Paradells, J. *Adapting AODV for IEEE 802.15. 4 mesh sensor networks: theoretical discussion and performance evaluation in a real environment* in *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks* (2006), 159–170.
11. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D. & Levis, P. *Collection tree protocol* in *Proceedings of the 7th ACM conference on embedded networked sensor systems* (2009), 1–14.
12. Dawson-Haggerty, S., Tavakoli, A. & Culler, D. *Hydro: A hybrid routing protocol for low-power and lossy networks* in *Proceedings of IEEE International conference Smart Grid Commun* (2010), 268–273.
13. Winter, T. *et al. RPL: IPv6 routing protocol for low-power and lossy networks* tech. rep. (IETF, 2012).
14. Levis, P., Clausen, T., Hui, J., Gnawali, O. & Ko, J. *The trickle algorithm* tech. rep. (IETF, 2011).
15. Ko, J. & Chang, M. MoMoRo: Providing Mobility Support for Low-Power Wireless Applications. *IEEE Systems Journal* **9,** 585–594 (2015).
16. Johnson, M. *et al. A comparative review of wireless sensor network mote technologies* in *Sensors, 2009 IEEE* (2009), 1439–1442.

17. Fotouhi, H., Zuniga, M., Alves, M., Koubâa, A. & Marrón, P. *Smart-hop: A reliable handoff mechanism for mobile wireless sensor networks* in *European Conference on Wireless Sensor Networks* (2012), 131–146.

18. Fotouhi, H., Moreira, D. & Alves, M. mRPL: Boosting mobility in the Internet of Things. *Ad Hoc Networks* **26,** 17–35 (2015).

19. Fotouhi, H., Moreira, D., Alves, M. & Yomsi, P. M. mRPL+: A mobility management framework in RPL/6LoWPAN. *Computer Communications* **104,** 34–54 (2017).

20. Cobârzan, C., Montavont, J. & Noel, T. *Integrating mobility in RPL* in *European conference on wireless sensor networks* (2015), 135–150.

21. Kuntz, R., Montavont, J. & Noël, T. Improving the medium access in highly mobile Wireless Sensor Networks. *Telecommunication Systems* **52,** 2437–2458 (2013).

22. Buettner, M., Yee, G. V., Anderson, E. & Han, R. *X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks* in *Proceedings of the 4th international conference on Embedded networked sensor systems* (2006), 307–320.

23. Ben Hamida, E., Chelius, G. & Gorce, J.-M. *On the complexity of an accurate and precise performance evaluation of wireless networks using simulations* in *Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems* (2008), 395–402.

24. Sneha, K. & Prasad, B. *An efficient hand-off optimization based RPL routing protocol for optimal route selection in mobility enabled LLNs* in *Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), 2016 International Conference on* (2016), 130–137.

25. Gara, F., Saad, L. B., Hamida, E. B., Tourancheau, B. & Ayed, R. B. *An adaptive timer for RPL to handle mobility in wireless sensor networks* in *Wireless Communications and Mobile Computing Conference (IWCMC), 2016 International* (2016), 678–683.

26. Somaa, F., El Korbi, I., Adjih, C. & Saidane, L. A. *A modified RPL for Wireless Sensor Networks with Bayesian inference mobility prediction* in *Wireless Communications and Mobile Computing Conference (IWCMC), 2016 International* (2016), 690–695.

27. Somaa, F., Adjih, C., El Korbi, I. & Saidane, L. A. *A Bayesian model for mobility prediction in wireless sensor networks* in *Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN), International Conference on* (2016), 1–7.

28. Barcelo, M., Correa, A., Vicario, J. L., Morell, A. & Vilajosana, X. Addressing mobility in RPL with position assisted metrics. *IEEE Sensors Journal* **16,** 2151–2161 (2016).

29. Ali, Q. I., Abdulmaowjod, A. & Mohammed, H. M. *Simulation & performance study of wireless sensor network (WSN) using MATLAB* in *2010 1st International Conference on Energy, Power and Control (EPC-IQ)* (2010), 307–314.

30. Cobarzan, C., Montavont, J. & Noel, T. *Analysis and performance evaluation of RPL under mobility* in *Computers and Communication (ISCC), 2014 IEEE Symposium on* (2014), 1–6.

31. El Korbi, I., Brahim, M. B., Adjih, C. & Saidane, L. A. *Mobility enhanced rpl for wireless sensor networks* in *Network of the Future (NOF), 2012 Third International Conference on the* (2012), 1–8.

32. Carels, D., De Poorter, E., Moerman, I. & Demeester, P. RPL mobility support for point-to-point traffic flows towards mobile nodes. *International Journal of Distributed Sensor Networks* **11,** 470349 (2015).
33. Duquennoy, S., Landsiedel, O. & Voigt, T. *Let the tree bloom: Scalable opportunistic routing with orpl* in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems* (2013), 2.
34. Johnson, D. B., Maltz, D. A., Broch, J., *et al.* DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Ad hoc networking* **5,** 139–172 (2001).
35. Liu, H., Yang, L. & Zhang, Y. Improved AODV routing protocol based on restricted broadcasting by communication zones in large-scale VANET. *Arabian Journal for Science and Engineering* **40,** 857–872 (2015).
36. Ferrari, F., Zimmerling, M., Thiele, L. & Saukh, O. *Efficient network flooding and time synchronization with glossy* in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on* (2011), 73–84.
37. Ferrari, F., Zimmerling, M., Mottola, L. & Thiele, L. *Low-power wireless bus* in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems* (2012), 1–14.
38. Zhong, X. & Liang, Y. Scalable Downward Routing for Wireless Sensor Networks and Internet of Things Actuation. *arXiv preprint arXiv:1802.03898* (2018).
39. Chatterjee, P. & Das, N. *Multiple sink deployment in multi-hop wireless sensor networks to enhance lifetime* in *Applications and Innovations in Mobile Computing (AIMoC), 2015* (2015), 48–54.
40. Dandekar, D. R. & Deshmukh, P. *Energy balancing multiple sink optimal deployment in multi-hop wireless sensor networks* in *Advance Computing Conference (IACC), 2013 IEEE 3rd International* (2013), 408–412.
41. Jain, T. K., Saini, D. S. & Bhooshan, S. V. Lifetime optimization of a multiple sink wireless sensor network through energy balancing. *journal of Sensors* **2015** (2015).
42. Peixoto, J. P. J. & Costa, D. G. Wireless visual sensor networks for smart city applications: A relevance-based approach for multiple sinks mobility. *Future Generation Computer Systems* **76,** 51–62 (2017).
43. Gaddour, O. *et al. Co-RPL: RPL routing for mobile low power wireless sensor networks using Corona mechanism* in *Industrial Embedded Systems (SIES), 2014 9th IEEE International Symposium on* (2014), 200–209.
44. Carels, D. *et al.* Support of multiple sinks via a virtual root for the RPL routing protocol. *EURASIP Journal on Wireless Communications and Networking* **2014,** 91 (2014).
45. Farooq, M. O., Sreenan, C. J., Brown, K. N. & Kunz, T. *RPL-based routing protocols for multi-sink wireless sensor networks* in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on* (2015), 452–459.
46. Murugan, K. & Pathan, A.-S. K. Prolonging the lifetime of wireless sensor networks using secondary sink nodes. *Telecommunication Systems* **62,** 347–361 (2016).
47. Safa, H., Moussa, M. & Artail, H. An energy efficient Genetic Algorithm based approach for sensor-to-sink binding in multi-sink wireless sensor networks. *Wireless networks* **20,** 177–196 (2014).

48. Jiang, H. & Sun, R. Energy optimized routing algorithm in multi-sink wireless sensor networks. *Appl. Math* **8,** 349–354 (2014).

49. Adewumi, O. G., Djouani, K. & Kurien, A. M. *RSSI based indoor and outdoor distance estimation for localization in WSN* in *Industrial Technology (ICIT), 2013 IEEE International Conference on* (2013), 1534–1539.

50. Misra, I. S. *Wireless communications and networks: 3G and beyond* (McGraw Hill Education (India) Pvt Ltd, 2013).

51. Dunkels, A., Gronvall, B. & Voigt, T. *Contiki-a lightweight and flexible operating system for tiny networked sensors* in *29th annual IEEE international conference on local computer networks* (2004), 455–462.

52. Mehmood, T. COOJA Network Simulator: Exploring the Infinite Possible Ways to Compute the Performance Metrics of IOT Based Smart Devices to Understand the Working of IOT Based Compression & Routing Protocols. *arXiv preprint arXiv:1712.08303* (2017).

53. Rappaport, T. S. Wireless Communications–Principles and Practice, (The Book End). *Microwave Journal* **45,** 128–129 (2002).

54. Bai, F. & Helmy, A. A survey of mobility models. *Wireless Adhoc Networks. University of Southern California, USA* **206,** 147 (2004).

55. Rahman, M. U., Alam, A. & Abbas, S. *Investigating the impacts of entity and group mobility models in MANETs* in *2016 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube)* (2016), 181–185.

56. Tall, H., Chalhoub, G. & Misson, M. *CoLBA: a Collaborative Load Balancing Algorithm to avoid queue overflow in WSNs* in *2015 IEEE International Conference on Data Science and Data Intensive Systems* (2015), 682–687.

57. Ancillotti, E., Bruno, R. & Conti, M. Reliable data delivery with the IETF routing protocol for low-power and lossy networks. *IEEE Transactions on Industrial Informatics* **10,** 1864–1877 (2014).

58. Chalhoub, G., Tall, H., Wang, J. & Misson, M. *DFTR: Dynamic Fault-Tolerant Routing Protocol for Convergecast WSNs* in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)* (2017), 1–7.