UNIVERSITY OF GOTHENBURG

Energy Efficient, High-speed Communication in WSNs

Master of Science Thesis in the Programme Computer Science

ATTILA NAGY

UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Göteborg, Sweden April, 2014

Energy Efficient, High-speed Communication in WSNs

ATTILA NAGY

Examiner: Olaf Landsiedel

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden April, 2014

MASTER OF SCIENCE THESIS IN COMPUTER SCIENCE

Energy Efficient, High-speed Communication in WSNs

ATTILA NAGY

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden April, 2014

Abstract

This thesis presents a set of feasible extensions for the low power, low delay ORW routing protocol. It introduces the capability of handling multiple concurrent bulk transfers including various application scenarios, e.g., cross traffic. The extensions added to the ORW protocol include a collision avoidance method applied beside the already existing, well functioning collision detection technique. This collision avoidance method reduced the overuse of the ORW protocol's collision detection method from a per packet basis to a per bulk transfer basis. This reduction in the frequency of collisions allowed a significant improvement in all the metrics in general, but especially in the transmission time and power consumption. Using the collision avoidance extension a bulk transfer was performed in a fraction of the time that would needed for a bulk transfer using the ORW base protocol, while the power consumed during this fraction of time was less accordingly. The second extension's purpose was to stabilize the EDC routing metric used by the ORW protocol to estimate the duty-cycles needed for a packet to reach the sink from the given node. This extension was relevant in certain scenarios with high intra-path interference stemming from the high number of hops in the paths between the source and the sink nodes. The third, and last, extension forbade the duty-cycling during a bulk transfer for the nodes that were involved in the given bulk transfer. This extension aimed to mitigate the performance degradation that occurred at the case when a given node terminated its duty-cycle notwithstanding that an other node was sending packets to this given node. By applying these extensions it was possible to reach an almost 500% increase in the throughput with less than 25% of the power consumption during a bulk transfer on the Indriya testbed.

Keywords: high-throughput, opportunistic routing, bulk transfer, WSN, TinyOS

Notations

WSN	Wireless Sensor Networks,
OR	Opportunistic Routing,
ExOR	Extreamly Opportunistic Routing,
MORE	MAC independent Opportunistic Routing,
ORW	Opportunistic Routing for Wireless Sensor Networks,
ORWE	Opportunistic Routing for Wireless Sensor Networks with Extensions,
EDC	Expected Duty Cycle,
PIP	Packets in Pipe,
LLH	Lossy Links, Low Power, High Throughput,
NUS	National University of Singapore

Contents

1 Introduction

This report presents the result of a thesis project in the field of Computer Science with the topic of opportunistic routing applicable for bulk transfers in WSNs. This chapter gives an overview to this project and its background, purpose and structure.

1.1 Motivation

In *wireless sensor networks*, most protocol stacks are designed for low data-rates. This is a widespread application scenario in WSNs and matches the limited resources of sensor nodes in terms of bandwidth and energy. However, there is a set of situations, in which we demand for high-speed bulk transfers: the energy efficient transport of large amounts of data through the resource constrained WSNs. Such scenarios, for example, include the distribution of OS updates and configurations, or the collection of raw measurement traces and logs from individual nodes. The purpose of the project was to provide a feasible solution for this problem along with a thorough evaluation covering a wide range of scenarios including intra-path interference, inter-path interference and concurrent bulk transfers. To achieve this goal, the project is based on an earlier developed, energy efficient, opportunistic routing protocol, *ORW*, for wireless sensor networks, described in detail in section 2.2. The aim of this project is to extend the ORW protocol with a capability of handling bulk transfers while keeping its appealing properties. Beside of this cutting edge protocol, for design and implementation the well-known TinyOS sensor network operating system was used.

1.2 Problem Statement

The base ORW protocol performed poorly in one particular scenario, which was bulk transfer. This performance degradation was reflected by the following measures:

- a drop in reliability,
- a greater number of duplicate packets,
- a high duty cycle period and CPU load, and
- an excessive transmission time on the application layer.

These measures were influenced by several factors, such as:

- the given topology,
- the benchmark,
- the number of executions of the measurement for the average calculation, and
- the underlying hardware used during the simulation.

Chapter 4 will present certain combinations of these factors with their corresponding measurement results.

1.3 Solution Overview

The solutions for the problems discussed in section 1.2 has the following three parts:

- busy-flag,
- EDC stabilization, and

- limiting duty cycling.

The busy-flag is the core of all the solution and the other two parts, the EDC stabilization and the duty cycling limitation, cannot be applied without it. This part of the solution aims to lower the probability of collisions in an inter-path interference intense topology by introducing a collision avoidance method. By using this collision avoidance method a node that is sending a packet from a bulk transfer only has to resolve a collision for the first packet from the given bulk transfer; the rest of the packets from the transfer will be only acknowledged by the node that forwards the first packet. All the other forwarder nodes that were eventually not allowed to forward the packet will end their duty cycle in case no other node attempts to send them a packet.

The second part of the solution aims to lower the fluctuation in the ORW protocol's routing metric. This routing metric, namely EDC, is used to estimate the number of expected duty cycles needed for a packet to reach the sink from a given node. In cases when a node sends a relatively high number of packets in a short period, this metric can be distorted. These cases can have various negative effects, e.g., extra hops in a packet's path, increase in power consumption, packet loss, and delays. The solution provided for this problem is to simply forbid any kind of EDC update during a bulk transfer.

The third part of the solution was dealing with the case when a duty cycle was terminated before all the bulk transfer packets were transferred to the sink. This problem generates delays in the network and therefore increases of the power consumption of the nodes that are used by the bulk transfer. The solution for this problem is similar to the solution for the second problem. The duty cycling for a node is only allowed when this node does not send any bulk packets. But if the node is actively sending bulk packets, then this node is not allowed to turn off its radio transceiver until all the packets were not sent from that bulk transfer.

The designs for these solutions are discussed in more detail in chapter 3.

1.4 Result Summary

The evaluation of the protocol extensions used two different benchmarks, a micro and a macro benchmark. The micro benchmark tested the performance of the extension protocol in certain isolated network scenarios using the Cooja simulation environment, while the macro benchmark was done on Indriya, a large scale, publicly available, low-cost testbed. The design of the extension protocol has the capability to perform multiple concurrent bulk transfers with cross traffic scenarios, while it has no effect on a non-bulk packet transfer.

The micro benchmark tested the following three scenarios:

1. inter-path interference,
2. intra-path interference,
3. concurrent bulk transfers.

The extension protocol managed to yield the best possible result in the inter-path interference scenario. That is all the metrics, covering reliability, transmission time, power consumption and transition count, stayed on a fix level by the increase of the number of possible forwarder nodes. Meanwhile, the base protocol suffered significant performance degradation in almost all of those metrics. In the intra-path interference scenario, the extension protocol made improvements on all the four core metrics mentioned above. For instance, the extension protocol consumed 10% of the power that was consumed by the base protocol with loosing only half of the packets that were lost by the base protocol. The last scenario in the micro benchmark, the concurrent bulk transfers scenario, showed that the extension protocol is capable of handle multiple concurrent bulk transfers even if their paths intersects each other. The evaluation using the Indriya testbed showed a 500% increase in the throughput in general with using 25% of the power that was consumed by the base protocol.

More data and evaluation of the design solutions are presented in chapter 4.

1.5 Outline

Chapter 2 gives a background on the given topic and terminology involving alternative solutions for high-throughput, both traditional and opportunistic routing methodologies, wireless sensor networks in general, and the routing protocol itself that was used during this project. Chapter 3 contains a thorough analysis of the problems within the base protocol, the scenarios where those problems are more likely to happen and the design of the extensions for the base protocol that aims to solve the problems identified during the analysis. Chapter 4 contains a detailed description about the metrics and topologies used during the measurements, the results of these measurements, and a discussion covering certain tendencies in the results, drawbacks, benefits and possible using areas. Finally, the last chapter, chapter 5, gives a summary of the project with suggestions for future improvements.

2 Background

This chapter provides the necessary background knowledge that is required for the understanding of the problem and the extension protocol introduced during the analysis of the next chapter, chapter 3. This chapter describes the components that were used during this project, the ORW protocol functioning as the base protocol, and several alternative solutions for high-throughput scenarios.

2.1 Components

2.1.1 Sensor Nodes

Sensors play a vital role in our daily life most of the time without revealing themselves to us. They can be found in several real life applications, e.g., bridges to avoid catastrophic infrastructure failures, control systems for farming, security systems and smart homes. These sensors are mostly placed in rough environment where changing batteries or hardware components are highly cumbersome processes. Therefore, energy-efficiency, remote access, size, robustness, durability, reliability and capability of autonomous operation are the key properties of such elements. Energy-efficiency can be gained by using 8 or 16 bits microcontrollers running at 1-10 megahertz with a few kilobytes of RAM instead of 32 or more bits CPUs and gigabytes of RAM. Remote access is granted by a low power radio transceiver that can send a few hundreds of kilobits per second. Their success in real life applications is granted by their low price, and the rapid development of embedded computing and wireless communication.[DP10; LG09]

As for the node's platform, a great set of platforms exist for WSN nodes such as: IRIS, MICAz, Imote2, Cricket and TelosB. For this project the TelosB platform was used shown by picture 2.1.

2.1.2 Wireless Sensor Networks

A WSN can consist of a number of sensor nodes in a large variety starting from a few nodes up to thousands of nodes. Numerous network architectures exist, but most of them follow the architecture depicted by figure 2.3, where multiple, self-organized, low power sensors form an ad-hoc mesh network with one designated *sink* node attached to a gateway. This gateway, on the other hand, has an internet connection, runs an embedded OS, and possess significantly higher computational power than the sensor nodes.

2.1.3 Traditional Unicast Routing in WSN

Traditional unicast routing in WSN can be divided into two separate phases. The first phase is to determine the next hop node based on routing metrics and link estimation. This process belongs to the routing layers. The second phase is part of the MAC layer and is to wait for the next hop node to wake up and receive the packet. The next subsection, subsection 2.1.4, will present several alternatives for traditional routing and introduce the protocol that was used during this project.

2.1.4 Opportunistic Routing Protocols

The basic difference between traditional unicast and opportunistic anycast routing is that the former first determines the next hop and then waits while this next hop wakes up and ready

Figure 2.1: TelosB sensor node. Taken from [adv13].

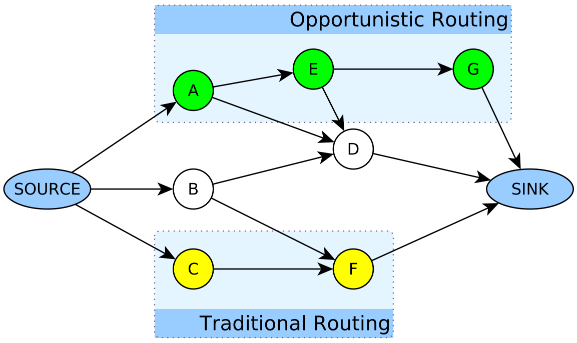


Figure 2.2: Routing Example

to receive the packet. On the other hand, the next hop in opportunistic routing can be any forwarder node that provides routing progress.

This difference is further explained by the following example depicted by a destination-oriented directed acyclic graph, figure 2.2. Node A is the first node that wakes up and, therefore, it receives the packet from the source in case of opportunistic routing. While node C, the node that has the best estimated link quality according to traditional routing, has its radio transceiver turn off. Node A will forward the packet to the first node that wakes up, in our case node E, meanwhile node C wakes up and can forward the packet to the estimated next forwarder, node F. Even though, in case of traditional routing the next hop is the $SINK$ reached by less number of hops, the opportunistic routing case yield lower delay due to the reduced waiting periods on each forwarder. Thus energy consumption is lower for this protocol.

Several versions of opportunistic routing protocols exist. A few among the most prominent ones are ExOR[BM05], MORE[Cha+07] and ORW[Lan+12]. They diverge in many aspects except one: all of them use anycast routing for addressing the next nodes in the routing path.

ExOR's greatest feature resides in its scheduler that helps to avoid duplicate packets in the network. Instead of choosing a fix sequential path, a list of forwarders are chosen by the source node based on their ETX value. This ETX information is collected periodically by link-state flooding, and designates a priority order for the forwarder nodes. The absence of duplicate packets is granted by the dedicated time slots for each forwarder.

On the other hand, MORE does not use a scheduler thereby providing spatial diversity and allowing all nodes to transmit at the same time. Since a special MORE header is attached to the packet along with the forwarder list, MORE introduces overhead to the network, with additional computation requirement and memory capacity on the nodes for network encoding and packets buffering.

Lastly, ORW incorporates several benefits of ExOR and MORE keeping their appealing features. Instead of adding a prioritized forwarder set to the packet header, in ORW a packet is forwarded by any node that gives routing progress. This approach yields two significant benefits:

1. a single EDC value in the packet header instead of lengthy forwarder set,
2. utilization of spurious and undiscovered neighbors.

Furthermore, by exploiting spatial diversity, ORW reduces duty-cycling on average by 50% and delays at least by 30% comparing to MORE and ExOR[Lan+12]. Thus, this project is based on this ORW protocol, and its design will be discussed in more details in section 2.2.

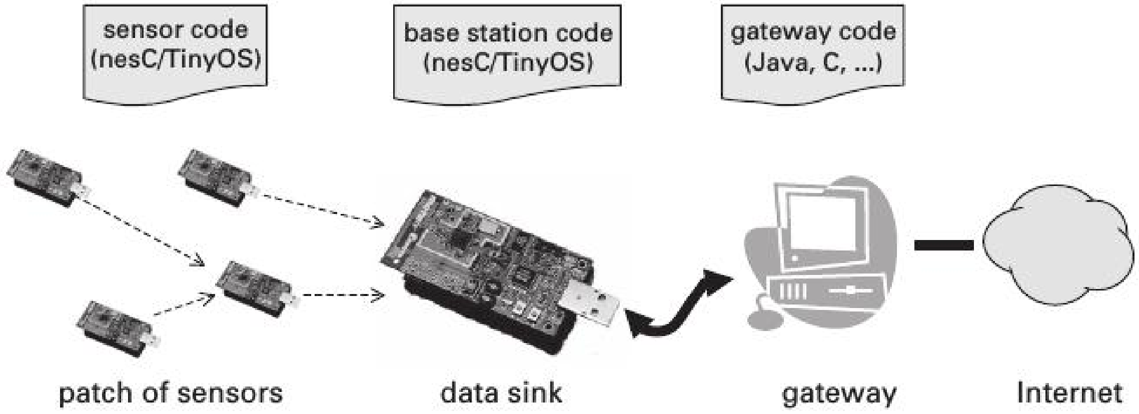

sensor code (nesC/TinyOS)
base station code (nesC/TinyOS)
gateway code (Java, C, ...)
patch of sensors
data sink
gateway
Internet

Figure 2.3: Network architecture of WSN. Taken from [LG09, p. 4].

2.1.5 TinyOS

TinyOS[LG09] is a de facto standard, open-source operating system with the following three main features:

component model: helps the developer to write simple, reusable code with high abstraction,

concurrent execution model: defines the interaction between the interrupt and non-interrupt context,

APIs, services, component libraries: suitable for writing new application code.

The component model includes implementation of basic components written in nesC language. Such component, can be, for instance, a user button, whose physical properties might vary depending on the given hardware architecture. To overcome the hindrance stemming from these variations, the component model gives a platform-independent, abstract interface for these components.

The concurrent execution model uses the split-phase strategy returning immediately after a request call with call-back functions for signalling the completeness of the required IO action. TinyOS uses tasks, which are lightweight deferred procedure calls, instead of threads. Tasks can be posted by any components and executed later, probably after a few milliseconds. Since the task execution cannot be preempted by an other task, this approach is data race free among tasks. However, a race condition can emerge in case of shared data between low-level interrupt code and a task, but this case is detected by the nesC compiler.

As for APIs, TinyOS has a wide set of APIs for common functionality, such as packet transmission or processing sensors data, for instance. Since TinyOS is a vivid project supported by an enthusiastic community, this set of APIs is continuously expanding and improving driven by the emergence of new hardwares and applications.

2.2 Design of ORW Protocol

The ORW protocol provides a low-power, low-delay packet transfer solution combining opportunistic routing with duty cycling for wireless sensor networks.

2.2.1 Expected Duty Cycle Wakeups

ORW uses *EDC*, a.k.a. expected duty cycle wakeups, network metric for calculating the expected number of node wakeup intervals until a packet reaches the sink node. The EDC value is calculated as the sum of three factors:

- the single hop EDC,
- the routing progress that neighbors offer divided by the sum of these neighbors' link quality,
- and ω, the cost of forwarding.

The single hop EDC is the inverse of the sum of the neighbors' link quality, and shows the number of time units needed to transmit a packet from the given node to the next hop.

$$EDC_i(S_i) = \frac{1}{\sum_{j \epsilon S_i} p_{ij}} + \frac{\sum_{j \epsilon S_i} p_{ij} EDC_j}{\sum_{j \epsilon S_i} p_{ij}} + \omega \tag{2.1}$$

2.2.2 Forwarder Sets

Determination of the *optimal forwarder set* plays a vital role in the ORW protocol, since this set guarantees that each packet sent from a node, acknowledged by an other node, progresses to the sink node hop by hop. Optimality is defined by the subset of forwarder neighbor nodes providing the lowest possible EDC value for the given node. As figure 2.4a shows, each node has an ordered table containing its neighbors with their EDC value. This table must be ordered since the number of the neighbors with the best EDC value determines the given node's forwarder set and thus its EDC value. The more the nodes in the forwarder set, the higher the chance that a node wakes up to receive the packet. However, not all node provides routing progress for the packet. Therefore, after a point the calculated EDC value for a node starts to increase by adding more neighbors to its forwarder set. This tendency is demonstrated by figure 2.4a.

Furthermore, the location of the forwarder set in the ORW protocol differs from other opportunistic routing protocols. In ORW the forwarder set is local to each node, while in ExOR and MORE, for instance, it is in the header of each packet. This design decision in the ORW protocol allows smaller packet headers, and therefore puts less overhead on network.

2.2.3 Protocol Stack

The protocol stack used for this project is depicted on figure 2.4b with the following three layers:

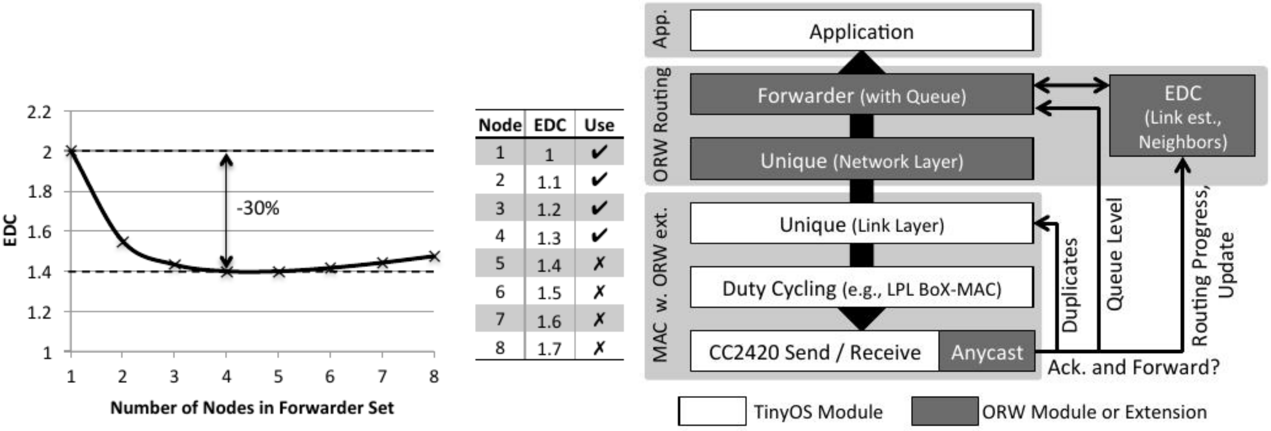
- MAC,
- ORW Routing,
- Application.

MAC Layer

The MAC layer is responsible for:

- transmission and reception of packets,
- duty cycling,
- acknowledging packets, and
- filtering of MAC layer duplicates.

The successful transmission of a packet on MAC layer depends on the number of awake neighbors ready to receive the given packet. For a neighbor node to be able to receive the



Node	EDC	Use
1	1	✔
2	1.1	✔
3	1.2	✔
4	1.3	✔
5	1.4	✗
6	1.5	✗
7	1.6	✗
8	1.7	✗

(a) Optimal forwarder set

(b) Protocol stack

Figure 2.4: Design of ORW protocol.
Taken from [Lan+12].

packet it must be awaken, provide routing progress for the given packet, and have at least one empty slot in its message queue. A node provides routing progress if its EDC value is lower than the sender's EDC, which was placed in the packet's header.

Every node except the sink operates in duty cycle mode. It simply means that the nodes turn on their radio receiver periodically to get all the packets being currently sent from their neighboring nodes. Emptying the message buffers are done by forwarding received packets both from other nodes or from the application layer. Moreover, by lowering the duty cycling period the probability for multiple nodes being awake at the same time is increasing leading to collision and thus retransmission on MAC layer.

The number of retransmissions and the acknowledgement of a packet depends on the number of nodes receiving the packet. A retransmission is triggered by receiving either multiple or zero acknowledgements on sender side. The former case can happen when there are more than one neighbor nodes that receive the packet, neither of them overhear the other's acknowledgement before it sends its own, and there was no previous retransmission with the given packet before. However, an extra factor is involved in the decision making depending on the number of retransmissions. This factor is a randomly generated extra condition that must be below a certain value given in percentage. The certain value is decreasing monotonously started from 50% and following the following general formula: $\frac{1}{n+1}$, where n is the number of retransmissions.

Receiving the same packet more than once on the same node leads to overhead on the network. Thus, filtering of these duplicate packets has a key role in the protocol.

There are two ways of receiving duplicate packets:

- on the MAC layer, or
- on the ORW layer.

Therefore, both layers have their own Unique sublayer with its own message buffer for tracking the incoming packets and providing a single instance of the given packet to the upper layer. Duplicates on MAC layer are mostly generated by retransmissions due to collisions. However, several scenarios exist that result in packets slipping through the MAC layer's Unique sublayer and only detected on the sink node by the ORW routing layer's Unique sublayer. One of the scenarios can be when one of the receiver nodes that caused the collision did not receive the retransmitted packet just the packet with the next sequence number. Therefore, this node cannot gain any information about the collision whatsoever, and forwards the packet as if there were no collision. Another scenario can be that one of the acknowledgements was not transmitted properly. Since the sender node only receives one acknowledgement, it assumes that there was no collision by sending the next packet which will be forwarded by both forwarder nodes. While there are several reasons for a packet loss, the most common reason stems from the noise range of an other arbitrary node. If one of the forwarder nodes F_1 is in the noise range, or can be even in the proper transmission range, of an other arbitrary node A, but an other forwarder node F_2 is not in this range, then this arbitrary node A's packet can jam the packets sent from other nodes to that forwarder F_1, but cannot jam the packets sent to F_2.

ORW Routing Layer

The ORW routing layer has the following roles:

- forwards unique packets to the application layer,
- stores the packets to be transmitted in a queue,
- puts the node's EDC in the packet header,
- removes duplicates that were not detected by the MAC layer,
- calculates the EDC value for the given node, and
- keeps information about the neighbors in a table.

The Unique sublayer detects duplicate packets that slipped through the MAC layer. These packets put more overhead to the overall network than the ones that were filtered out on lower layers, since these packets might travel through the whole network and detected only by the sink node.

The Forwarder sublayer queues the unique packet received either from the Application layer or from other nodes. Furthermore, this sublayer forwards the packets to the Application layer on the sink node, and puts the given node's EDC value in the packet header's destination field.

As for the EDC sublayer, its roles are to keep the neighbors' EDC and link quality in a table, update these values on events such as transmitting or receiving a packet, and to calculate the given node's EDC value that is going to be sent to the neighbor nodes in the packet header.

Application Layer

The Application layer is where the user defined messages are being created at a non-sink node in the network and eventually received on the sink node. The maximum transmission unit for application layer messages can be calculated with the following formula:

$$MTU_{App} = MTU_{CC2420} - sizeof(Header_{ORW}) \tag{2.2}$$

The default MTU_{CC2420} is 28 bytes, but this value can be set higher by a compilation flag. The size of the ORW header, as it is shown in section 2.2.4, is 5 bytes.

2.2.4 ORW Header

The ORW header has four parts:

- a TTL counter (7 bits),
- a flag for pull request (1 bit),
- a sequence number (2 bytes), and
- a node number (2 bytes).

The TTL counter guarantees that no packet stays in the network circulating forever. At every hop this counter is incremented by one and checked if it does not exceed a predefined value. If it does exceed this value, then the packet is not forwarded and thus removed from the network. The pull request flag urges the receiver node to send out dummy packets to discover its neighbors. Each message sent from the Application layer is identified by its node wise unique sequence number and its node number.

2.3 Related Work

This section introduces three high-throughput alternatives of opportunistic routing. The first solution is PIP [Ram+10], a connection-oriented, multi-hop, multi-channel, TDMA-based MAC protocol. The second solution, called Flush [Kim+07], on the other hand, shows a CSMA-based protocol applying a rate-control algorithm along with end-to-end acknowledgements. While the last solution is the LLH [DÖD11] protocol that provides burst forwarding with low power consumption under heavy interference.

2.3.1 Packets in Pipe Protocol

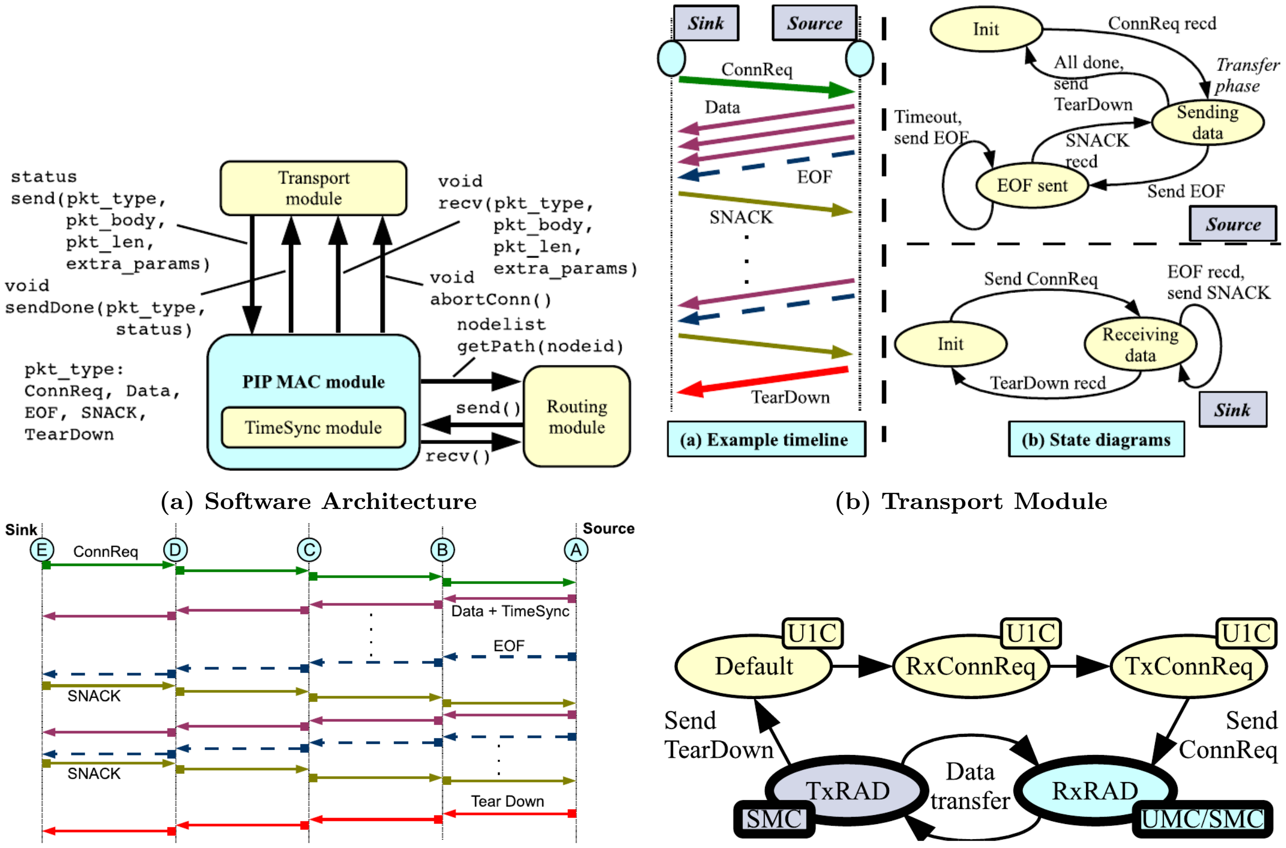
The Packets in Pipe protocol, shortly PIP, is a viable solution for bulk transfer scenarios in WSN. This solution is a combination of other already well-known methodologies and design solutions in the field of computer networks. The reason why it is remarkable in its field is due to its unique combinations of the following set of design features:

- multi-hop, connection oriented,
- uses TDMA-based channel access method,
- uses multiple radio channels,
- centrally controlled network.

Furthermore, above all these features that were listed above, PIP has several appealing properties, such as:

- the number of hops slightly influences the throughput,
- robust design for wireless errors,
- no need for flow-control,
- small queue size is sufficient for good performance.

As for the reasons behind those four design decisions, PIP uses TDMA instead of CSMA due to CSMA's high intra-path interference under bulk transfer. Secondly, choosing TDMA opens the possibility for multi-channel data transmission. The throughput gain by this method, obviously, depends on the underlaying protocol, since those protocols determines the number of usable channels accessible by PIP. The higher the number of these channels, the higher the gain in performance. However, channel switch produces a noticeable, but rather negligible, overhead on each node. Thirdly, PIP has a centralized architecture due to the fact that all WSNs already possess a special node, the sink node. Therefore, extending the role of this node would not change anything on the structure of the network. Furthermore, while in general a distributed architecture can have a simpler design, in case of protocols aiming for high-throughput such an architecture would require a far more complicated design. Lastly, PIP uses a connection-based design approach to let the nodes in the data flow path know about their time-slot and channel they are allowed to use for transmission. This information is given to the nodes during the process of establishing the connection and used until the tear down of the connection.



(a) Software Architecture

(b) Transport Module

(c) PIP MAC Example

(d) PIP MAC Module

Figure 2.5: Design of Packets in Pipe Protocol. Taken from [Ram+10].

Figure 2.5a shows the PIP protocol architecture. PIP uses three different modules, the Transport module to handle end-to-end reliability, the PIP MAC module with the TimeSync submodule for building the connection for the Forwarder module, and the Routing module for finding the best path to the source. The Transport module uses the state machines presented on figure 2.5b to transmitting data from source the *Sink*. The connection is always requested by the sink node via sending a *ConnReq* message. The *Source* node, in turn, replies to this request by sending the bulk data, possibly in multiple data packets, ended by a special packet, the End-of-File packet. When the *Sink* receives this EOF packet, it replies to the *Source* with *SNACK*, a packet telling it which *Data* packets did not arrive and should be retransmitted. When all the missing packets were retransmitted the *Source* node initiates the tear down of the connection.

Figure 2.5d shows the state machine for the PIP MAC module. This module is the responsible for providing the connection-oriented interface used by the Transport module and for time-synchronization among the *Source* and all the other nodes in the path to the *Sink*. The state machine has three states:

1. the unsynchronized, single-channel mode (U1C),
2. the time-synchronized, multi-channel mode (SMC), and
3. the unsynchronized, multi-channel mode (UMC).

Figure 2.5c depicts the flow of an example execution of the PIP MAC protocol on a multi-hop network. By default, all the nodes are in *Default* state. Since there is no ongoing data transmission, the single-channel mode is used. This mode uses a reserved channel that is not

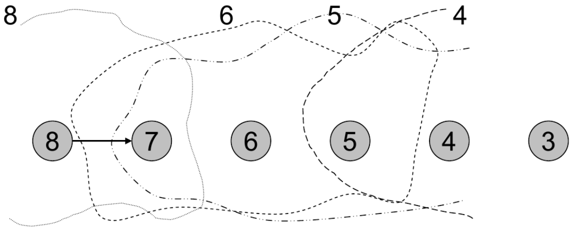


Figure 2.6: Packet transfer from node 8 to 7 interfering with transfer from node 5 to 4. Taken from [Kim+07]

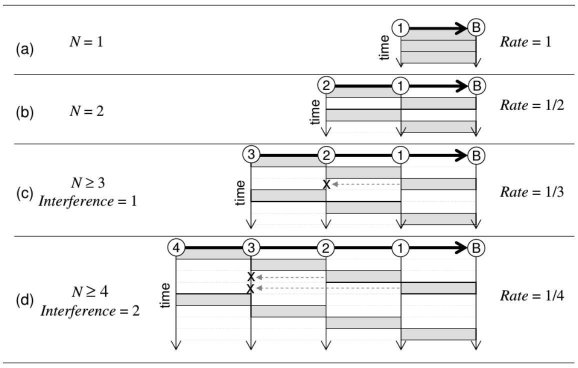


Figure 2.7: Maximum transmission rate without collision. Taken from [Kim+07]

used by any nodes being in SMC or UMC state. When the *Sink* requests a connection to the *Source*, it switches to the *RxRAD* state in mode UMC and stays in that mode until it does not receive the first *Data* packet. After the first *Data* packet, the *Sink* switches to the SMC mode allowing the use of multiple channels for data transmission. The reason why the *Sink* first has to be in the UMC state and not in the SMC is that the *Source* node is piggybacking time synchronize date to the *Sink* by the *Data* packet. Until the first *Data* packet has not arrived, the *Sink* node is in an unsynchronized state. The *Source* node is in *TxRAD* state during the bulk data transfer in mode SMC; it skips the UMC mode. The return to the *Default* state is initiated by the *Source* node by sending the *TearDown* packet.

For the multi-channel transmission a schedule must be determined by the network that is followed by all the nodes. The determination of this schedule is done by the *Sink* node, and spread across the network by the *ConnReq* packet's PIP MAC header. A schedule contains the exact channel and slot that a given node is allowed to use for transmission. When the *TearDown* packet is received by an intermediate node, this node forgets the channel and time slot that was given to it earlier and switches back to U1C mode waiting for a new connection to be requested by the *Sink* node.

2.3.2 Flush Protocol

The Flush protocol uses a similar approach to the PIP in a sense that there can be only one bulk transfer ongoing and that is requested by the sink node. Furthermore, Flush uses pipelining as well as PIP, with some limitation stemming from intra-path interference. Since Flush uses CSMA/CA media access control protocol, instead of TDMA as PIP does, it can only apply pipelining where the packets sent by successor nodes do not interfere with the packet sent by a predecessor node of the same node. Figure 2.6 depicts an example for interference, where node 8 has to wait with its transmission to node 7 until all the nodes that are closer to the sink, node 5, 6, 7, finish their transmission. To orchestrate this process a dynamic rate control technique is used in Flush. The maximum sending rate depends on two properties:

- the number of hops between the source and sink node, and
- the interference ranges.

Figure 2.7 shows examples for calculating the maximum rate. When $N = 1$, the sender does not have any kind of limitation on its transmission. When there is a node between the sender and the sink there is still no interference, but nodes can only transmit with rate 1/2. Examples (c) and (d) show the maximum transmission rate for models with four and five nodes interfering with each other. By exceeding the maximum transmission rate a node can receive packets from its predecessor and successor node at the same time leading to collision of the packets.

The level of interference used during the calculation of the transmission rate is determined by overhearing the packets sent by the successor nodes; a.k.a. snooping. However, each node N above its transmission range has a noise range as well where nodes cannot snoop packets from this node N but packets from node N might cause interference for the nodes in its noise range. As stated in [Kim+07], the Flush protocol accepts some level of interference derived from this noise range by saying: ”While Flush’s interference estimation is not perfect, its window of inaccuracy is narrow”.[Kim+07, p. 5]

2.3.3 Lossy Links, Low Power, High Throughput Protocol

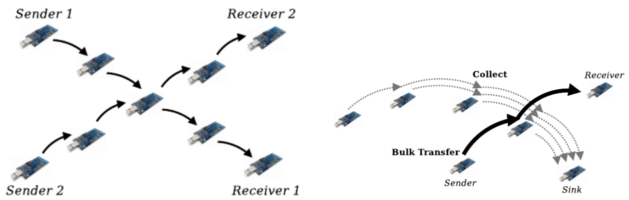
The Lossy Links, Low Power, High Throughput protocol, shortly LLH, is an alternative that is more similar to the protocol that was used during this project. It follows a generic packet forwarding technique, called burst forwarding, combining the merit of low-power and high-throughput characteristics available in other state-of-the-art protocols, such as PIP and Flush.

The following four features are common between the LLH and the ORWE protocols:

- duty cycling with low-power listening,
- high resistance against both intra-path and inter-path interference,
- possibility for multiple concurrent bulk transfers,
- using CSMA MAC protocol.

These features make the LLH and ORWE protocols applicable for certain application scenarios that neither PIP nor Flush are capable to perform. Just a few of these scenarios can be the cross traffic when the paths of two bulk transfers intersect each other, figure 2.8a, or the scenario when the non-bulk traffic sent by multiple concurrent applications is intersected by an other node’s bulk transfer, figure 2.8b.

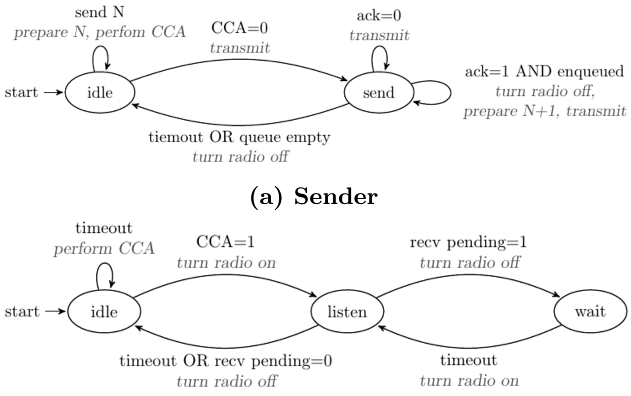
However, there are a few differences between the two protocols as well. For instance, the LLH protocol uses inter-packet sleeping which basically means that between two consecutive packets sent by a given node, retransmissions are excluded, this given node turns off its radio transceiver to spare energy. This optimization does not initiate any kind of delays to the transmission process if the transceiver is turned back on before it is used. A typical case when this optimization can be applied is when a packet was successfully transmitted, and the sender node has to load the new packet to its radio chip. The two state machines on figure 2.9 give an insight into the transmission process showing the cases when the radio module is turned on and off.



(a) Cross-traffic

(b) Concurrent applications

Figure 2.8: Applications areas for LLH. Taken from [DÖD11].



(a) Sender

(b) Receiver

Figure 2.9: State machines in the LLH protocol. Taken from [DÖD11].

3 Design

This chapter first analyses the problems residing in the base ORW protocol, section 3.1, and then gives a thorough description about the proposed solutions for these problems in section 3.2.

3.1 Analysis

The base ORW protocol performed poorly in one particular scenario, which was bulk transfer. This performance degradation was visible in the following measures:

- reliability,
- number of duplicates on MAC layer,
- duty cycle period and CPU load,
- application layer transmission time, and
- EDC stability.

The low performance in the above measures was traced back to the following three problems:

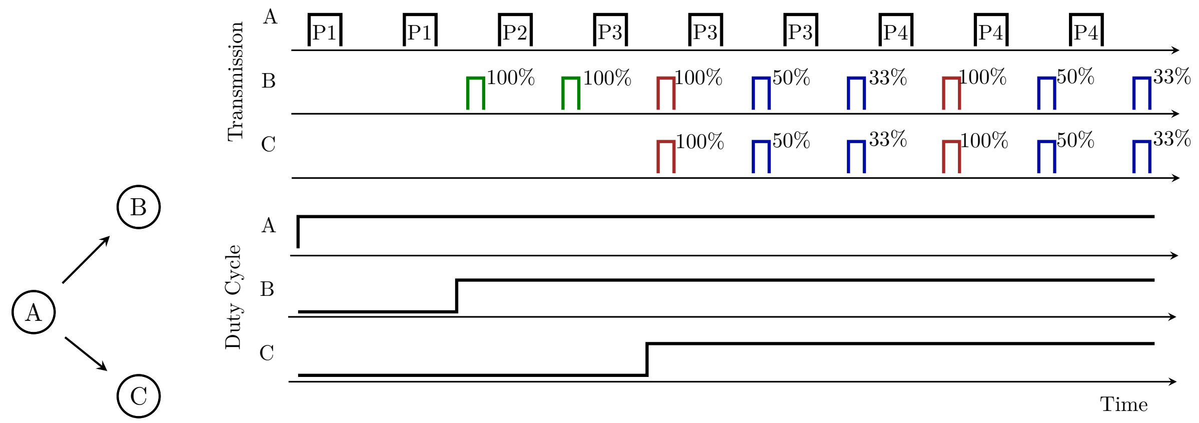
- collisions,
- EDC fluctuation,
- early termination of duty cycles.

These problems will be described in this section, while the designs of the solutions for these problems are covered in section 3.2.

3.1.1 Collision Problem

The most common problem happens when one node starts to send a high number of packets without any delay between them and the transmission of these packets between the two nodes does not end before another node wakes up. In that case, the second node that just recently woke up starts to acknowledge the packets as well as the first node, leading to collisions between these two receiver nodes. The problem with this strategy is that in this situation collisions must be resolved packet wise, as depicted on figures 3.1 including the topology, figure 3.1a, and transmissions with duty cycle of each node, figure 3.1b. Thus if there are a lot of packets to be sent, then it results in an *excessive transmission time* between these two nodes. The *high duty cycle period* can be explained by the increase in transmission time. Furthermore, the more the collisions in the acknowledgements, the more the retransmission on MAC layer. Due to this rule, *more duplicate packets* are caught on the MAC layer's Unique sublayer resulting in an increase in *CPU load*.

Figure 3.1b shows the probability of collisions in scenarios when there are multiple forwarder nodes during bulk transfer. When node B wakes up it can acknowledge packets $P1$ and $P2$ sent by node A without any difficulties. However, after a certain point when node C wakes up and starts to acknowledge the packets from node A, the acknowledgements from the two nodes collide which can be only resolved by retransmission. That is the reason why node A sent the packet $P3$ three times; first, both nodes B and C acknowledged it since they did not know anything about each other and assumed that there is no other receiver. When they received the same packet again then they could conclude that either there is another receiver somewhere or acknowledgement was lost and node A did not receive it. For avoiding deadlock situations the receiver nodes are compelled to act in accordance with their former conclusion, even though an acknowledgement loss can lead to extra retransmissions by this strategy.



(a) Topology **(b) Probability of collisions for topology in figure 3.1a using the base protocol.**

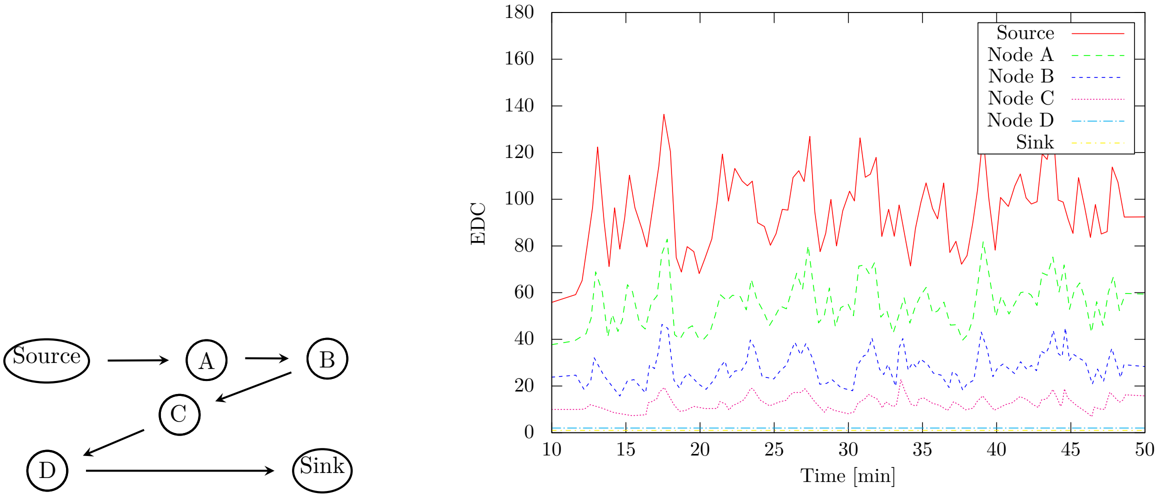
Figure 3.1: A collision prone situation

When an acknowledgement loss and the forwarder node receives the packet again it assumes that the packet was resent due to collision. Therefore, the forwarder node will lower the chances of acknowledging this packet to 50%. However, if the packet was resent due to acknowledgement loss, then retransmissions after the acknowledgement loss were unnecessary and only initiated delays and duplicates on MAC layer. Such a case happens with a higher probability in topologies with lossy links. Furthermore, while acknowledgement loss only leads to a single event of retransmission, having two nodes that actually collide result in a number of colliding packets. The worst problem with this is that when a node receives packets it does not end its duty cycle, and therefore this collision situation will be present until the end of bulk transfer. Thus, nodes A and B have to resolve their collision for packet $P4$ as well and for the other packets that are yet to come after $P4$. The probability of acknowledgement is depending on the number of retransmissions of the given packet and calculated as follows: $\frac{1}{n+1}$, where n is the number of retransmissions.

3.1.2 EDC Fluctuation Problem

Since one node receives a high number of packets in a short period from the same node, all the links' quality to other nodes get reduced in this short period. These changes in the link qualities affect not just the given nodes EDC, but all the other nodes that send their packets via this given node. Therefore, further the node from the sink node, greater the *fluctuation in its EDC value.*

A simple example for this phenomenon is shown by figures 3.2a and 3.2b. While the former figure presents a basic topology prone to EDC fluctuation, the latter gives a plot for each node's EDC value during bulk transfer using that basic ORW topology. Furthermore, the EDC calculation for the same topology is provided by the equations from 3.2 to 3.7 using the methods described in section 2.2.1 and simplified in the current section as equation 3.1.



(a) Topology prone to EDC fluctuation.

(b) EDC plot during bulk transfer for topology in figure 3.2a.

Figure 3.2: EDC fluctuation using the base protocol.

$$EDC_n = \frac{1}{p_{n,n+1}} + \frac{p_{n,n+1} * EDC_{n+1}}{p_{n,n+1}} + \omega = \frac{1}{p_{n,n+1}} + EDC_{n+1} + \omega \tag{3.1}$$

$$EDC_{Sink} = \omega \tag{3.2}$$

$$EDC_D = \frac{1}{p_{D,Sink}} + EDC_{Sink} + \omega = \frac{1}{p_{D,Sink}} + 2 * \omega \tag{3.3}$$

$$EDC_C = \frac{1}{p_{C,D}} + EDC_D + \omega = \frac{1}{p_{C,D}} + \frac{1}{p_{D,Sink}} + 3 * \omega \tag{3.4}$$

$$EDC_B = \frac{1}{p_{B,C}} + EDC_C + \omega = \frac{1}{p_{B,C}} + \frac{1}{p_{C,D}} + \frac{1}{p_{D,Sink}} + 4 * \omega \tag{3.5}$$

$$EDC_A = \frac{1}{p_{A,B}} + EDC_B + \omega = \frac{1}{p_{A,B}} + \frac{1}{p_{B,C}} + \frac{1}{p_{C,D}} + \frac{1}{p_{D,Sink}} + 5 * \omega \tag{3.6}$$

$$EDC_{Source} = \frac{1}{p_{Source,A}} + EDC_A + \omega = \frac{1}{p_{Source,A}} + \frac{1}{p_{A,B}} + \frac{1}{p_{B,C}} + \frac{1}{p_{C,D}} + \frac{1}{p_{D,Sink}} + 6 * \omega \tag{3.7}$$

Finally, fluctuations in EDC lead to situations where nodes that do not provide actual routing progress become part of the forwarder set. It happens when the jump in the EDC of node B, for instance, exceeds the EDC of node A. In that case node A can acknowledge the packets from node B, and, since node A's unique message buffer probably still contains those packets, they are identified as duplicates and, therefore, not forwarded. Eventually, this chain of events results in a *drop in reliability* on application layer since these packets were both removed from the forwarder buffer of node B due to their successful transmission and not saved in any buffer of node A.

3.1.3 Early termination of duty cycles

Duty cycling during a bulk transfer can lead to situations where nodes that are heavily used turn off their radio receiver. This problem highly increases the packets' transmission time and

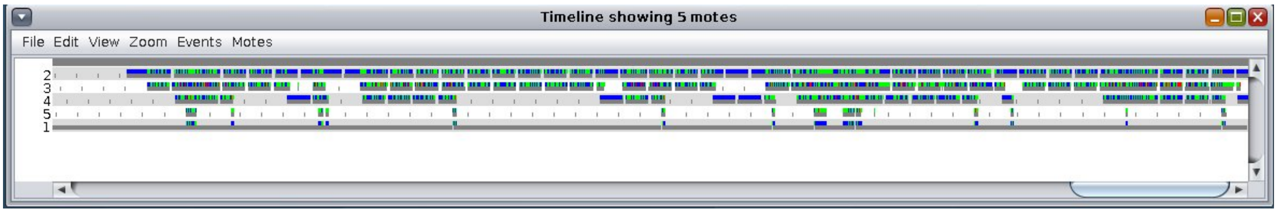


Figure 3.3: Duty cycling during bulk transfer

the power consumption of the whole network.

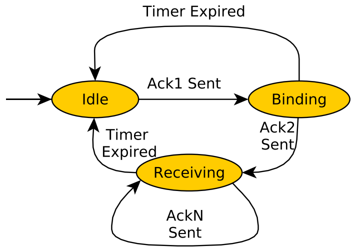
The root of this problem resides in the property of the base protocol that ends the duty cycle of the node when its message buffer gets empty. This is not a problem in low traffic situations, since there are no significant number of packets on the neighboring nodes waiting to be sent to the given node. However, it can be a problem in a protocol where all the bulk packets has the same path.

However, during bulk transfer a node with turn off radio transceiver can initiate delays on multiple nodes depending on the given topology. At topology 3.2a, for instance, if node $D's$ radio is off, then all the nodes further from the *Sink* including the *Source* wait for node D to wake up; otherwise, these nodes cannot reach the *Sink*.

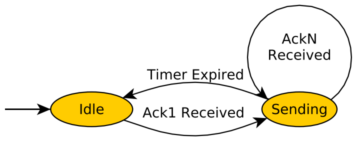
Figure 3.3 shows the timeline of a concrete execution of the 3.2a topology with three forwarder nodes: 3, 4, 5. The *Source* is the node with the id 2, while the *Sink* is the node with id 1. The dark grey color indicates that the radio transceiver for a given node is on. As it can be seen from the figure, the *Sink* has its radio always turned on while the rest of the nodes turn on their radio periodically for a short period to see if somebody is sending packets. If they do, then the node first receives the packet and then empties its message buffer by turning to sending mode. The blue color shows that a node is currently transmitting a packet, and the green color shows that a node is receiving. What can be seen from the figure is that the first node the starts to send packets is the *Source* node, obviously, and then each node that is one hop closer to the *Sink* wakes up and forward the packet. But, the last forwarder node, node 5, switches from receiving mode to sending mode while node 4 sends packets. The problem is that node 5 terminates its duty cycle when its message buffer get empty and therefore node 4 has to wait until node 5 wakes up again. The same problem happens with node 3 and node 4 when node 4 empties its message buffer. Eventually, all the nodes that are further from the *Sink* than a given node N depend on that node N and cannot end their duty cycle until all the nodes that are closer to the *Sink* than node N wake up and forward their packets to the *Sink* giving space for the new packets from node N in their message buffer. This is reflected by the node wise aggregated duty cycle period, since the closer the node to the *Sink* the less time it waits for other nodes, and the less time it waits for other nodes, the less time it spends in duty cycling.

3.2 Design of ORW Extensions

This section gives a thorough description about the design of the extension protocol focusing on the problems that were pointed out in section 3.1. The first of these problems was the packet wise race condition during collisions which led to excessive delays and power consumption in the network. The solution for this problem is covered in subsection 3.2.1. The second problem stems from the EDC instability during bulk transfer resulting in reliability degradation in certain topologies. The design of the solution for this problem is described in subsection 3.2.2. The last problem, discussed in subsection 3.2.3, included nodes terminating their duty cycles



(a) Receiver side



(b) Sender side

Figure 3.4: State machines

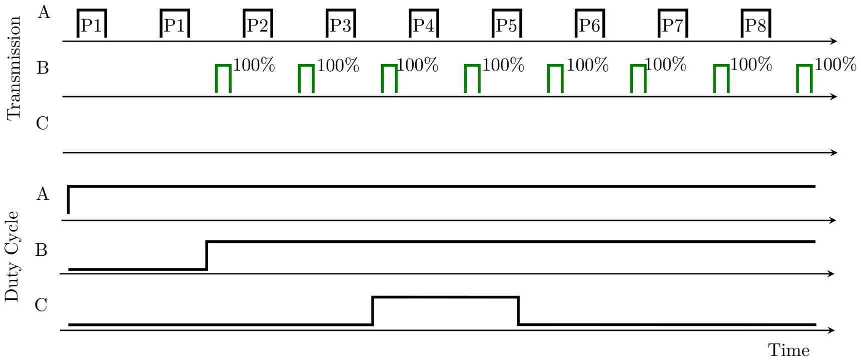


Figure 3.5: Probability of collisions for topology on figure 3.1a using the extension protocol.

too early and therefore initiating additional delays and power consumption in the network.

3.2.1 Collision Avoidance

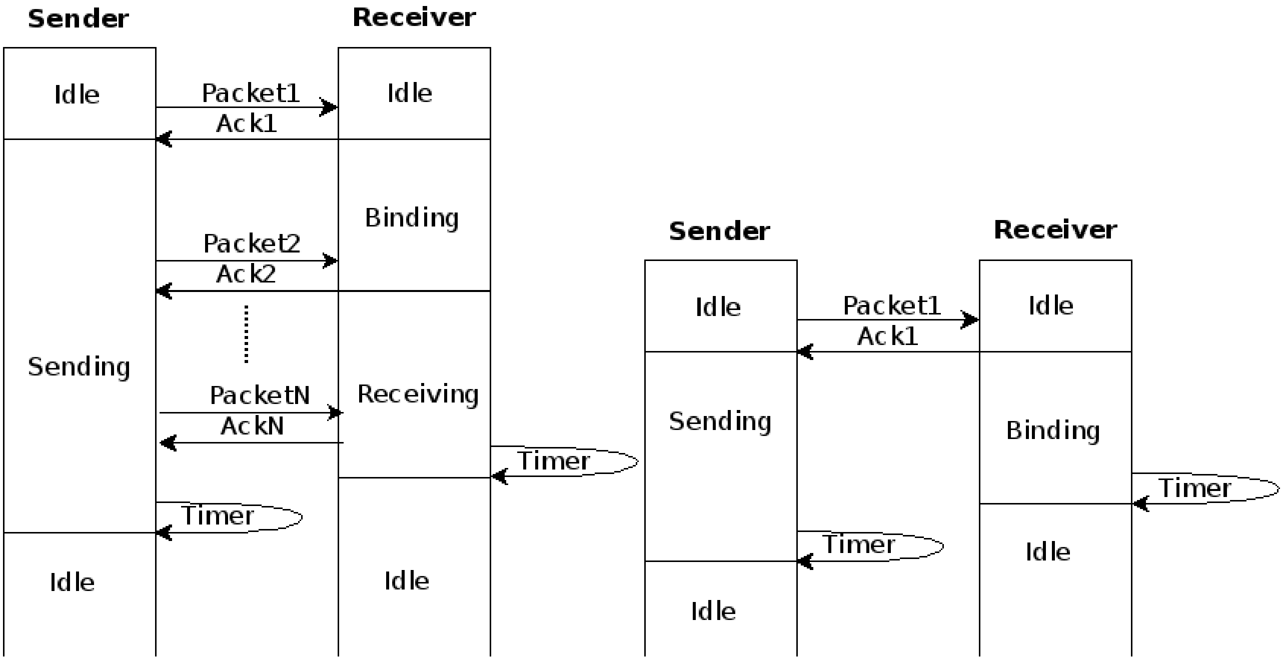
The design of the collision avoidance in the extension protocol covered the most part of the project. It is divided into two parts: sender and receiver part. The sender part contains the design for setting a special flag, the *busy-flag*, when there is a node that can forward the given node's packets. The receiver part keeps track of the sender nodes from which the given node is allowed to forward packets with busy-flag. If a node receives a packet with busy-flag but never received a packet from that node before, then this node can conclude that the packet was intended for an other forwarder node, and therefore shall neither acknowledge nor forward that packet. After certain amount of time this second forwarder node that does not forward packets will end its duty cycle eventually. A node to be able to accept packets from multiple sender nodes must store the state of a state machine, described in more details later in this subsection, separately for each of its neighbors that sent at least one packet to this given node in the past.

This chain of events is depicted on figure 3.5. Initially node A sends packet $P1$, but there is no node ready to receive it. The first node that wakes up is node B and since node A was sending the packet without busy-flag, node B can acknowledge and forward it. The next packets, $P2$, $P3$, and so on, will be sent with busy-flag allowing node B to draw the conclusion that there were no collision when it acknowledge $P1$. When node B acknowledges one packet with a sequence number N and receives the next packet with the sequence number $N + 1$, then this node can always conclude that there was no collision and it is allowed to forward packet from that node until the end of bulk transfer. Node C when wakes up it only turns on its radio for a short period to check if there is any node that is sending packet that it can acknowledge. Having all the packets sent from node A with busy-flag on, node C is not allowed to acknowledge them, and therefore terminates its duty cycle.

Figure 3.4a shows the state machine of the packet receiving process. Initially, every node starts from the *Idle* state. It is due to the fact that a node could not receive any packet while its power was off. After the first packet was received from a node that has higher EDC, meaning from a node that is further from the sink as the actual node, and the packet has no busy-flag, the node sends an acknowledgement to that packet and switches to *Binding* state. From *Binding* state a node has two options to proceed. First, if it receives the next packet with sequence number $N + 1$ from the same node, where N was the sequence number of the previous packet, then it can acknowledge the packet and switch to *Receiving* state. The second option is not to receive any packet from the same node in a certain amount of time t and resetting the receiver state back to *Idle* for that sender node in the actual node. It must be emphasized that the state is reseted only for that sender node, since each neighbor has its own state machine in a given node allowing to receive packets with busy-flag from multiple nodes at the same time. In *Receiving* state a node can accept any number of packets from the same sender node. The node will stay in the *Receiving* state until the gap between two received packets does not exceed that certain amount of time t that was used earlier for resetting from the *Binding* state. When this time t is exceeded in the *Receiving* state, then the receiver state machine is reseted back to *Idle* state. This period of t was chosen to be equal to the local wake up of a node used in the base ORW protocol, which is 2048 ms, due to synchronization reasons.

Figure 3.4b, on the other hand, shows the state machine of the sender side, which is fairly simple compared to the receiver side state machine. It has only two states, the *Idle* and the *Sending* state. Each node starts from the *Idle* state when it is powered on, for a similar reason why the receiver state machine starts in the *Idle* state; the node assumes that it did not send any packets while it was turned off. When it receives the first acknowledge for its packet being sent without any collision, it switches to *Sending* state and stays there for the rest of the bulk transfer or until the forwarder node stops acknowledging its packets. The latter case can happen when a forwarder node's message buffer gets full and thus it has to start forwarding packets instead of receiving but there is no other node for that forwarder node to forward those packets. In this case the optimal solution for the actual sender node is to send those packets to an other forwarder, if such a node exists, instead of waiting for the first forwarder node to finish emptying its message buffer. However, if there were collisions due to multiple acknowledgement, then the collision avoidance strategy is the same as it was in the base protocol. That is, the sender keeps retransmitting the packet until there was only one node sending acknowledgement. The forwarder nodes reduce their chances to acknowledge the packet at each retransmission. The chance for acknowledgement is the following: $\frac{1}{n+1}$, where n is the number of retransmissions. When the collision is resolved, the sender node will send the next packet with the new sequence number allowing only the sole node that sent the last acknowledgement to accept and forward the rest of the bulk transfer.

Figure 3.6a shows the states from the two state machines in action during a bulk transfer. Both *Sender* and *Receiver* nodes start from the *Idle* state, where they are allowed to send and receive packet to and from any node. The *Sender* first sends $Packet1$ without the busy-flag. The *Receiver* decides that it can accept this packet and therefore send the first acknowledgement, $Ack1$, to the *Sender*, and switches to *Binding* state by this action. Since *Sender* receives only one acknowledgement, it can switch to *Sending* state without any problem and send the next packet, $Packet2$. Node *Receiver* receiving this packet can conclude that there was no collision with other nodes when it sent $Ack1$ and therefore allowed to switch to *Receiver* mode and send $Ack2$ to the *Sender* node. Being the *Sender* in *Sending* state and the *Receiver* in *Receiving* state, the *Sender* can send any number of packets to the *Receiver* without any collision with other nodes. There are two ways of exiting these states. The first is that the *Receiver*'s message buffer gets full and therefore it has to start sending packets to its own forwarder nodes to free slots in its buffer for the packet from the *Sender* node. If this process takes too long, then a timer expires and resets the *Receiver*'s receiving state from the *Sender*



(a) Bulk transfer **(b) Not bulk transfer**

Figure 3.6: Sequence diagrams

node to *Idle*. The other possible way of ending a bulk transfer is that the *Sender* run out of packets to be sent and thus expires its timer clearing its sender state to *Idle*.

Figure 3.6b, on the other hand, shows the sequence diagram of a scenario when one packet was sent. This scenario is not a bulk transfer scenario, but it is still relevant to show that there is no performance degradation compared to the base protocol. The only difference between this scenario and a bulk transfer scenario is that the *Receiver* node resets its receiver state machine to *Idle* from the *Binding* state instead of *Receiving* state. The reason why this strategy does not have a lower performance than the base protocol, is due the fact that in the worst case scenario the extension protocol behaves exactly the same way as the base protocol does. The worst case scenario is when the gap between two packets is just slightly higher than the timer's period putting the highest number of collision on the *Sender* that is possible. In this case, both designs have to resolve the collision packet wise.

3.2.2 EDC Stabilization

The EDC fluctuation problem has a fairly simple solution. This solution is to simply prohibit the EDC update of a node if it is involved in a bulk transfer. A more interesting question would be how this involvement is defined. A node is involved in at least one bulk transfer if any the following three conditions is true:

- Is the given node in *Sending* state?
- Is the given node in *Receiving* state from any node?
- Has the given node overheard any packet with busy-flag from other nodes in a certain period?

If any of the above questions if true, then the EDC update for the given node is prohibited until the next attempt for update where the above conditions are checked again.

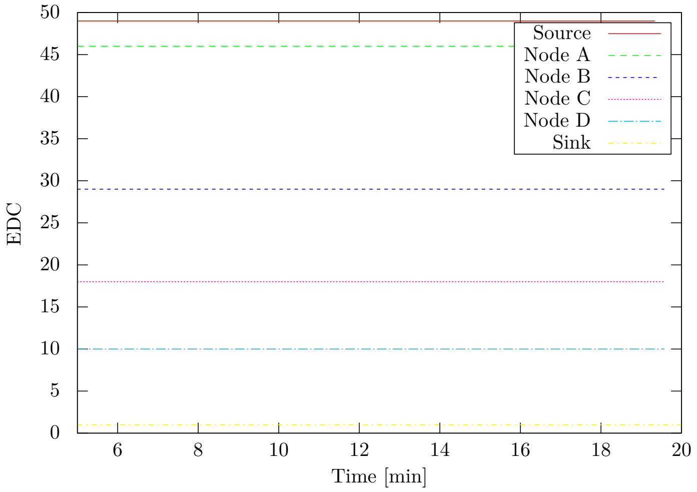


Figure 3.7: EDC fluctuation using the extension protocol for topology in figure 3.2a

The result for such strategy is show by the graphs in figure 3.7 using the topology from figure 3.2a. Each node has a well balanced EDC value at the beginning of a bulk transfer and this balance is kept until the end of the transfer. This strategy prevents loop-backs in packet paths and thus eliminates reliability drops stemming from the loop-backs.

3.2.3 Limitation on Duty Cycling

Duty cycling is only applied when a given node's sender state machine is not in *Sending* state. When it is in *Sendig* state, the radio transmitter is always on for that given node. This limitation is essential in the extension protocol to prevent nodes to end their duty cycle too early. In case of early termination of receiver node's duty cycle, the node that sends the packets has to wait until the forwarder node wakes up. This unnecessary waiting time initiates extra delays and power consumption into the network.

Figure 3.8 shows a snapshot from the Cooja simulator from a test execution when there was no duty cycling during the bulk transfer. As it can be seen, this absence of duty cycling allowed the transmission of a bulk transfer to be executed without major gaps in the execution timeline. For this execution the same topology was used as for figure 3.3 which is similar to the topology in figure 3.2a with the only difference that it has three forwarder nodes.

The duration of a bulk transfer, however, for this design solution has a different definition than it had in subsection 3.2.2, at the EDC stabilization solution design. In this case, the only condition that is need to be true is whether the given node's sender state machine is in *Sending* sending state. If it is in *Sending* state, then the given node is not allowed to end its duty cycle until it runs out of packets to be sent and resets its sender state machine to *Idle* state. It is important to point out that, even though the condition covers only the sender part, it has effect on both the sending and the receiving mechanism.

However, this solution is not a remedy without boundaries. The higher the number of the forwarders in the chain the more the time that is needed for these nodes to wake up at the beginning and start to forward the bulk packets. If the chain of forwarders is too long, then by the time that the first bulk packets arrive to the last forwarder node, the one that is nearest to the sink, the first forwarder's timer has already expired and reseted its sender state machine's state to *Idle* and thus terminated its duty cycle. This is exactly the same problem that we wanted to solve just on a topology with more forwarders. Increasing the timer period for the

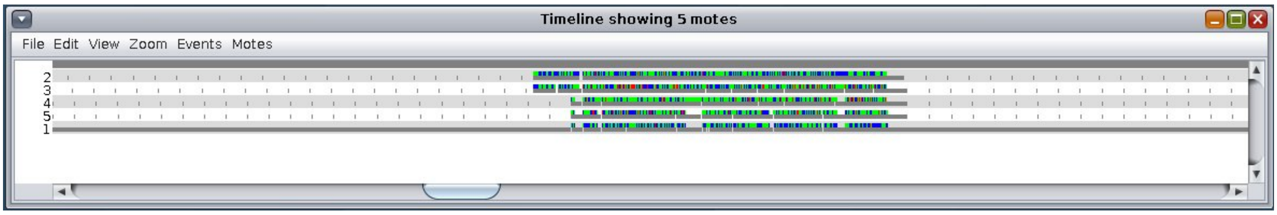


Figure 3.8: No duty cycling during bulk transfer

Sending state could mitigate this problem, but it would introduce additional delays and power consumption to the whole network for both bulk and non-bulk scenarios as well. Therefore, this solution solves the early termination of duty cycle problem for small scale topologies, but not applicable for topologies having more than four forwarder nodes in a chain.

In this chapter we discussed the three major problems with the base protocol using an illustration for the collision situations, EDC plots and snapshot of the timeline plug-in from the Cooja simulator. Furthermore, for each of these problems there was provided a solution in this very same chapter using the same type of figures giving a platform for comparison of the two designs in certain aspects relevant for the given problem. However, the comparison given in this chapter only presents the difference in a functional point of view and not in a sense of performance. The performance evaluation is presented in the next chapter, chapter 4.

4 Evaluation

This chapter describes two types of benchmarks, the microbenchmark in section 4.3 and the macrobenchmark in section 4.4. Furthermore, the metrics and the combinations of solutions that were used during the evaluation of those benchmarks are presented in sections 4.1 and 4.2 respectively. For the microbenchmark a cutting edge WSN simulation environment, Cooja, was used, while for macrobenchmark a three-dimensional wireless sensor network deployed across three floors of the National University of Singapore was applied. The testing during the development phase was done on the microbenchmark including a thorough performance measurement on various topologies and comparison with the base ORW protocol. Lastly, the macrobenchmark served as a platform for the evaluation on a greater scale and verification of the predicted behavior from the previous chapter, chapter 3.

4.1 Metrics

This section introduces the metrics that were used during the evaluation of both benchmarks. There are two types of metrics:

1. the one whose purpose is to show the performance improvements provided by the extensions, and
2. the one that shows that the problems discussed in chapter 3 are valid and present in the base protocol.

From all the metrics used in this project only two, the EDC fluctuation and the duplicate count, belong to type 2; all the rest were used to compare the base protocol with its extensions.

The used metrics are the followings:

Reliability: Reliability R_i shows the percentage of the successfully transmitted packets $P_{i,sink}$ on the application layer from source i to the sink during a bulk transfer:

$$R_i = \frac{\#P_{i,sink}}{\#P_i} \tag{4.1}$$

If there were multiple sources in the network, then the reliability R is the average of all the source nodes' reliability:

$$R = \frac{\sum R_i}{\#sources} \tag{4.2}$$

Power consumption: Power consumption PC is the summarized duty cycle period DC of each node i that it spends on sending and receiving data packets:

$$PC_i = \sum DC_{i,data} \tag{4.3}$$

$$PC = \sum PC_i \tag{4.4}$$

In other words, the time spent in idle state is excluded from the power consumption calculation.

Transmission time: Transmission time TT_i is the period between the arrival time on the sink for the last bulk packet sent by source node i and the sent time for the first packet from the same bulk transfer on the source node i:

$$TT_i = T_{P_{last},node_{sink},received} - T_{P_{1st},node_i,sent} \tag{4.5}$$

In case this last bulk packet arrives multiple times on the sink node, the arrival time that is used during the transmission time calculation is always the packet that is received the earliest. The transmission time TT is a summarized value of TT_i from all the source nodes:

$$TT = \sum T_i \tag{4.6}$$

Transition Count: Transition count TX simply counts all the packets that were sent on the Forwarder layer in the whole network including all the retransmissions and Forwarder layer duplicates.

Transmission Rate: Transmission rate TR_i is the ratio between the number of packets $P_{i,sink}$ that were successfully transmitted from the given source node i to the sink in a bulk transfer and the time TT_i that this transmission took.

$$TR_i = \frac{\#P_{i,sink}}{TT_i} \tag{4.7}$$

Delay: Delay D_i shows the summation of the set of periods between the time a packet P was sent on the source node i and the time it was received on the sink. On the other hand, metric D does not make any differentiation among packets from different source nodes; it calculates the average delay for all the packets P_n that was sent in the whole network during bulk transfer. The calculations for these metrics are the following:

$$D_i = \sum (T_{P_{n,node_i,sent}-T_{P_{n,sink,received}}}) \tag{4.8}$$

$$D = \frac{\sum D_i}{\#P_n} \tag{4.9}$$

Hop count: The number of hops HC_i that a packet needs in order to reach the sink node is dependant on the location of its source node i. During the evaluation of this report only the average of the hops from all the bulk packets sent by a given source node was used.

Duplicate count: Since there are two types of duplicate packets, the one that is caught on the MAC layer, DP_{MAC}, and the other one that is caught on the ORW layer, DP_{ORW}, they are counted separately. However, there is no node wise duplicate counting due to the fact that the density of duplicates across the network is not homogeneous and therefore the nodes cannot be compared, just the network itself.

EDC fluctuation: The EDC fluctuation of a node i is tantamount to the standard deviation σ of this given node's EDC values during bulk transfer. The standard deviation is calculated as follows:

$$\mu = E[X] \tag{4.10}$$

$$\sigma = \sqrt{E[(X-\mu)^2]} \tag{4.11}$$

$$= \sqrt{E[X^2] + E[(-2\mu X)] + E[\mu^2]} \tag{4.12}$$

$$= \sqrt{E[X^2] - 2\mu E[X] + \mu^2} \tag{4.13}$$

$$= \sqrt{E[X^2] - 2\mu^2 + \mu^2} \tag{4.14}$$

$$= \sqrt{E[X^2] - \mu^2} \tag{4.15}$$

$$= \sqrt{E[X^2] - (E[X^2])} \tag{4.16}$$

Table 4.1: Design combinations applied during evaluation

Design	busy-flag	EDC stabilization	limited duty cycling
ORW	X	X	X
ORWE	√	√	√
ORWE-BF	√	X	X
ORWE-EDC	√	√	X
ORWE-DC	√	X	√

The μ is the expected mean value of the random value X; that is, in our case, the updated EDC value of the given node i. Therefore, our final equation is:

$$EDCfluctuation_i = \sqrt{\frac{\sum(EDC_i^2)}{\#EDC_i} - \left(\frac{\sum EDC_i}{\#EDC_i}\right)^2} \tag{4.17}$$

4.2 Design Combinations

Table 4.1 shows the combinations of solutions from section 3.2 used during the evaluation process in this chapter. The ORWE design uses all the three solutions, and therefore aims to provide overall the best performance improvement from all the designs. However, in certain scenarios one of the other designs might give slightly better result due to the fact that that design was tailored for that given scenario only.

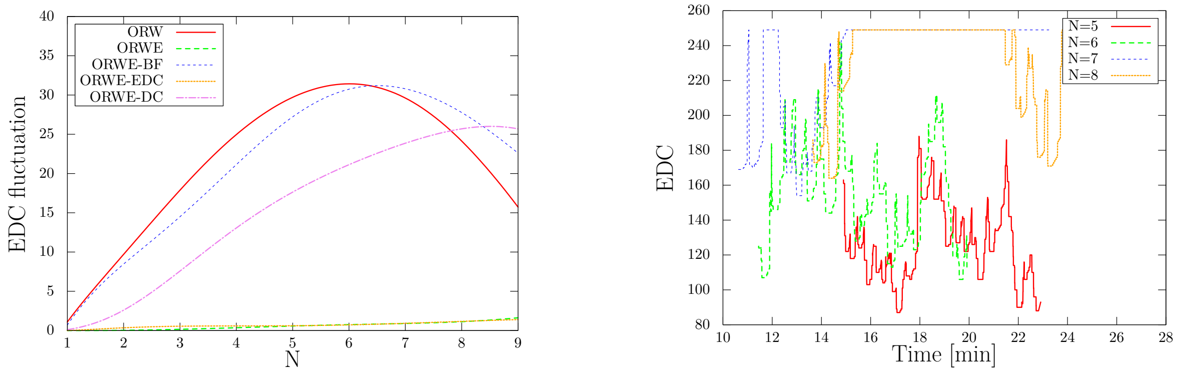
The ORW-BF design includes the basic busy-flag idea without any EDC stabilization and limitation on the duty cycling. Without this basic design the other two designs, the ORW-EDC and the ORW-DC, would not be usable, since those two designs depends on the bulk transfer mode, which is defined by the busy-flag states of a node's sender and receiver state machines. Therefore, the designs ORWE-EDC and ORWE-DC are the combination of two solutions, the busy-flag's combination with the EDC stabilization, and the busy-flag's combination with the duty cycle limitation.

4.3 Microbenchmark

The microbenchmark covers most of the work that was done during this project. It involves the assessment of the base protocol, the testing of the extensions, and provides a final evaluation exposing the limitations of the Cooja simulator on high performance situations. The final evaluation, presented in subsection 4.3.2, covers three types of scenarios:

1. inter-path interference,
2. intra-path interference,
3. concurrent bulk transfers.

The inter-path interference type aims to show the improvements on topologies where packets have multiple possible ways to get from the source node to the sink thus producing interference among these multiple paths. While, in the case of intra-path interference, there is only one possible path for all the packets sent from the source node to get to the sink. The interference that this type aims to test resides inside the path of a packet. Since the topologies used for this type allows only one path for all the bulk packets to traverse, it is possible to reference to a node that is closer to the source node than a given node as the predecessor node of that given node and reference to a node that is closer to the sink as a successor node. Using this terminology the intra-path interference can be defined as an interference between a predecessor



(a) EDC fluctuation on Source: Drop in fluctuation due to fake stability when there are more then 6 forwarders.

(b) EDC for *Source* node using ORW design: EDC that reaches the ceiling gives fake stability.

Figure 4.1: Problems with EDC

P and successor S nodes of a given node N. In more details, the packets sent by node P to node N can be jammed by node S if the radio ranges, including the noise range, of both nodes, P and S, cover node N. The purpose of the last type, the concurrent bulk transfers type, is to show the performance differences between the five designs in a scenario when multiple nodes send bulk transfer concurrently producing an interference intense network.

As for the testing methodology used for producing the results presented in this section, it uses the results of ten test executions per data point from different test seeds and produces graphs from the mean value of these executions. However, the plot containing a node's EDC values covers only one simulation, since its purpose is to demonstrate certain scenario instead of providing statistical data. For the parallel execution of the measurement scripts the free software *GNU parallel* [Tan11] was used.

4.3.1 The ORW Protocol

The first step in the evaluation process during the project was to justify the reasoning that was presented in the previous chapter, chapter 3. For this purpose, correlations must have been shown between the performance degradation under bulk transfer and the number of duplicate packets in the network along with the nodes' EDC stability using the base protocol. To pinpoint these correlations, a collision intense topology was chosen to generate duplicates both on MAC and ORW Routing layer. Figure 4.2 shows the number of duplicate packets on both MAC and ORW layers in the function of the number of forwarder nodes using the topology from figure 4.4a. The graphs from figure 4.5 show the effects of this increase in the number of duplicates inflicted on the core metrics. The transmission time of the bulk packets, for instance, increases by 2 seconds by every extra forwarder node connected parallel with each other. In a general form, it means $(n + 1) * 2$ second transmission time for 1000 packets, where n is the number of forwarder nodes connected parallel. This increase can be explained by the collisions and retransmissions that are necessary side-effects of duplicate packets. Similar tendency can be seen in the power consumption and transition count metrics originating from the same source. The higher the number of forwarder nodes in the topology from figure 4.4a, the higher the power consumption and the transition count. The reliability, on the other hand, does not suffer from any kind of negative effects in the base protocol due to its wisely devised collision detection strategy.

The second problem to be justified was the correlation between the EDC fluctuation and the above mentioned four metrics: reliability, power consumption, transmission time and transition

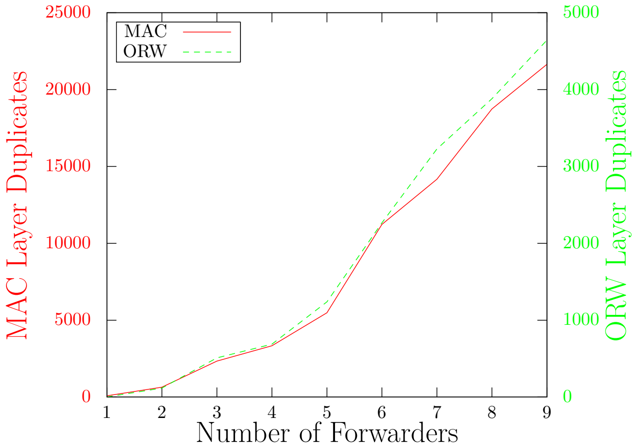


Figure 4.2: Duplicates of ORW design using topology 4.4a.

count. To increase the EDC fluctuation in the network, the topology, used also for intra-path interference testing, from figure 4.4b, already presented earlier in section 3.1, was used. In this type of topology the degree of the EDC fluctuation depends on the number of forwarder nodes chained between the source and sink nodes. The further the node from the sink, the greater the fluctuation. Figure 4.1a shows the *Source* node's EDC fluctuation in the function of number of forwarders. The figure shows a latent improvement in the EDC stability when the number of forwarder nodes are greater than 6. The reason for this fake improvement can be clearly seen on graph 4.1b. By increasing the number of forwarders, the average EDC value of the *Source* node is increasing as well. However, there is an upper bound for the possible EDC values which should not be reached by any node in the network. As the figure shows, this ceiling value is more likely to be reached and used in case of higher number of forwarders leading to a lower standard deviation in the EDC. Graphs from figure 4.6 show the effect made by EDC fluctuation on the core metrics. The reliability drops drastically by the increase of EDC fluctuation. This problem stems from the temporary loops in the EDC of two neighboring nodes allowing packets to travel back in path and be recognized as duplicates without letting the sender node know about it and therefore resulting in packet loss. This phenomenon was discussed in more details in the chapter 3. All the transmission time, the transition count and the power consumption increase significantly faster than in the previous inter-path interference case. The increase in the transmission time is reasonable, since each packet requires more hops to reach the *Sink* node from the *Source*. Consequently, more nodes have to forward a single packet to reach the *Sink* node resulting in the increase in power consumption and transition count.

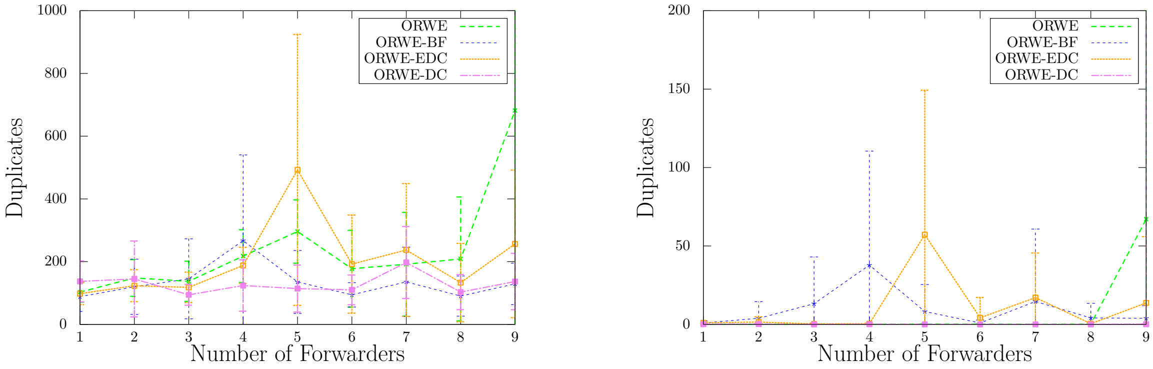
This subsection showed that the problems in the base protocol that were described in section 3.1 are valid and lead to significant performance degradation. The next subsection will evaluate each of the designs from section 4.2 and compares them with the base protocol.

4.3.2 Extensions for ORW

A group of extensions provides solutions for the performance degradation stemming from the EDC fluctuation, the excessive number of duplicate packets and the duty cycling during bulk transfer. This subsection shows the gained performance improvements of the extension protocol by comparing the core metrics on three types of topologies. The first two types, figures 4.4a and 4.4b, were introduced in subsection 4.3.1, while the third type, figure 4.4c, is similar to the first except it contains multiple source nodes. As for the core metrics, all the four metrics from the previous subsection, subsection 4.3.1, were included in the evaluation of the extension protocol. Figure 4.5 shows the four core metrics in the function of the number of forwarder nodes using the topology from figure 4.4a. This topology is relevant since it compares the new and old designs in a collision prone, inter-path interference intense situation. Its aim is to show the effectiveness of the solutions given for one of the core problems, namely the high number of duplicate packets. Furthermore, the number of duplicate packets for the old ORW design, figure 4.2, and the new ORWE designs, figures 4.3, are presented in separate figures due to the substantial difference in magnitude between those two groups. As for the EDC fluctuation, graphs from figure 4.6 compares the performance of the designs in the function of number of forwarder nodes placed in chain instead of parallel position forming a network with high intra-path interference. Figure 4.4b shows this type of topology. The more the forwarder node in the chain, the higher the fluctuation in their EDC value. The same topology from figure 4.4b with the same graphs from figure 4.6 were used for the evaluation of the duty cycling during bulk transfer problem. The third type of topology with the multiple source nodes on figure 4.4c shows the designs in a case when the number of source nodes concurrently sending bulk packets exceeds the number of forwarders. This topology formulates a bottleneck scenario in the sense that the number of forwarder nodes are not able to serve the performance demand generated by the source nodes creating a special collision critical situation. The core metrics for this topology is shown by the graphs from figure 4.7.

The two figures from 4.3 show that the number of duplicates on both MAC and ORW layers were reduced to a static level, where the increasing number of parallel forwarder nodes does not influence the number of duplicates in the network. Comparing to the graphs in figure 4.2 with 9 parallel forwarder nodes, for instance, the number of duplicates in the ORWE-DC design is less the 1% of the duplicates from the base ORW. On the top of that, the numbers of ORW layer duplicates were reduced close to zero in all the ORWE designs. The reason why the ORWE-EDC design with five forwarder nodes produced a relatively high standard deviation can be explained by the case where one test execution with an extremely high result value spoils the average of a consistent set of results. Therefore, the exclusion of that particular execution from the set of results is reasonable. However, the result of this execution is included in the dataset used for the 4.3 figures to highlight on the fact that the ORWE designs still have a low probability of generating a moderate number of duplicates. Nonetheless, the number of these duplicates are still not comparable to the number produced by the ORW design.

The graphs in figures 4.5 show that neither of the designs suffer loss in their reliability from inter-path interference. The basic ORW protocol seems to produce slightly worst reliability compared to any of the other four designs, but it is less than 1%, which means less than 10 packets loss in average from the 1000 bulk packets. The other three core metrics, on the other hand, show significant differences among the five designs. The ORWE group of designs was not affected by the increasing number of available parallel forwarder nodes N. Due to the busy-flag solution these designs choose one forwarder node and use that one node for the rest of the bulk transfer. This is the reason why the transmission time and transition count stayed on a static level independently from the change of N. However, there is a minor power consumption growth in the function of N for ORWE and ORWE-EDC. The reason for this growth can be found in the transition count graph. The ORWE-DC and ORWE-BF graphs has a suspiciously low transition count around 1500, while logically, since at least two nodes must send each packet at least once, the transition count cannot be lower than 2000. This is only possible if some of the logs were lost during the test execution. This suspicion was proven to be right by finding around 500 packets that were received on the sink node but never sent from any of the forwarders. The sole reason for these missing logs is that the EDC update during bulk transfer



(a) MAC layer duplicates **(b) ORW layer duplicates**

Figure 4.3: Duplicates of each ORWE designs: No significant increase in the number of duplicates.

can lower the logging intensity incurred on a forwarder node. While without the update the forwarder node might forward packets fast enough to empty the logging system's message pool ignoring requests for logging. Additional test executions with ORWE-BF and ORWE-DC using loggings limited only for the transition count and power consumption metrics show similar results to ORWE and ORWE-EDC.

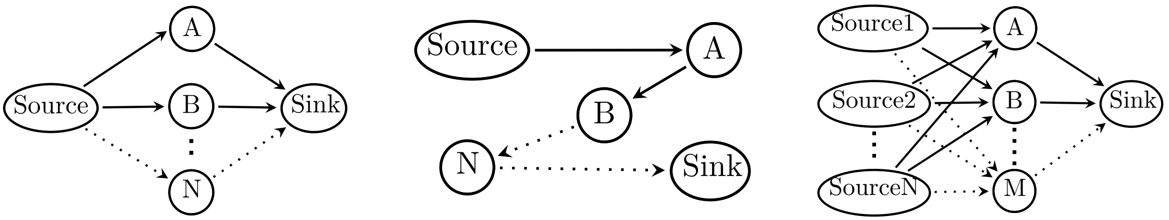
As for the intra-path interference testing graphs from figures 4.6, it is evident that the ORWE design gives the best performance in most of the core metrics. These graphs give an assessment for solutions of both the EDC fluctuation problem and the problem of the early termination of duty cycles during a bulk transfer. During this assessment the topology from figure 4.4b was used. Since ORWE-EDC and ORWE-DC contain the solutions for these two problems, they should show performance improvements in this given type of topology compared to the base ORW design. This expectation is satisfied in most of the metrics, but not in reliability, where ORWE-EDC has the steepest drop in the reliability. The underlying reason for that lurks in the nature of the simulation environment, described in details in subsection 4.3.3, stemming from its limitations under high CPU load per node. Both transmission time and power consumption, on the other hand, show the expected result. Both ORWE-EDC and ORWE-DC perform better than ORW, but the best performance is given by ORWE, the combination of those two. In the transition count metric, there are no significant differences in the curves of the five designs due to the fact that there is only one possible path for all the bulk packets – assuming there are no loop-backs in the packets' path. The differences between the curves only come from the retransmissions.

Lastly, the graphs from the concurrent bulk transfers scenario show substantial distinction among the types of the five designs, even though the standard deviation of the data points in these graphs are significantly higher reducing the weight of the conclusions draw from them. The main conclusion from the four metrics is that the ORWE definitely has better performance than ORW in case of multiple concurrent bulk transfers which is best reflected in the power consumption and transmission time metrics. As for reliability, all the designs, but especially ORW, have very high overlapping standard deviations in reliability. Therefore, a conclusion drawn from the reliability graph would be entirely meaningless.

4.3.3 Limits of Simulation

This section gives an insight to the circumstances around the problem in the Cooja simulation environment providing explanation for the performance degradation presented in figure 4.6.

In figure 4.6a, all designs have the same characteristics. That is, the reliability drops



(a) Inter-path interference **(b) Intra-path interference** **(c) Concurrent bulk transfers**

Figure 4.4: Topologies used during the evaluation of the three types of scenarios

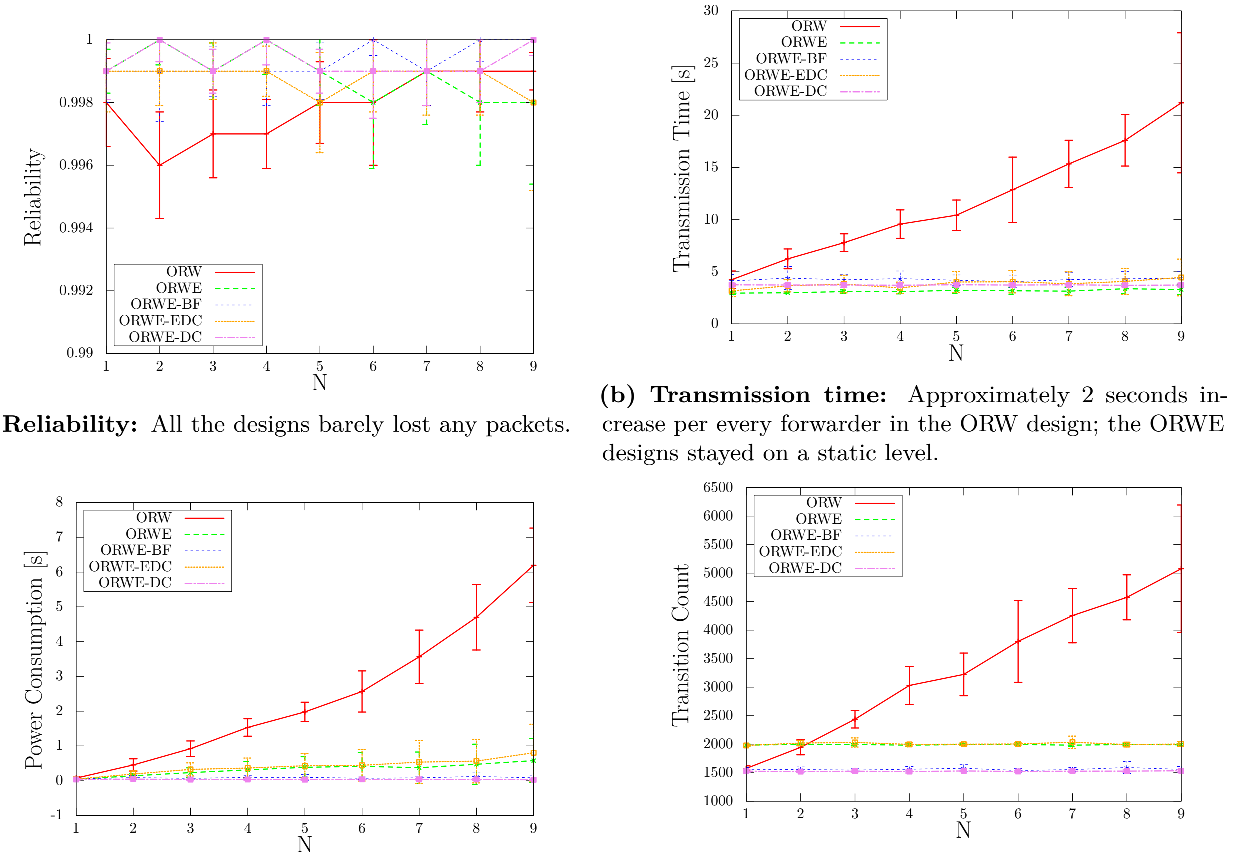
constantly from low number of forwarders to higher number of forwarders regardless to the actual design. This similarity indicates the probable situation where the actual differences between the designs were not to be shown by the graphs due to the performance cut stemming from the simulator's behavior under high CPU load per node. Since the hardware unit responsible for sending fast acknowledgement is simulated as well as everything else in the Cooja simulator, the acknowledgements are always sent with a greater delay in the simulation environment than by the real hardware nodes. This problem is shown by the snapshots from the Cooja simulator in figures 4.8 and 4.9. Figures 4.8 show information about a correctly sent packet and about an incorrectly sent packet with their acknowledgements using Cooja's RadioLogger plug-in. The MAC layer sequence numbers are marked with red color in the *Data* field. In the correct case, figure 4.8a, the sequence number of the acknowledgement, *C5*, matches with the sequence number sent by node 3. Node 4 can acknowledge node 3's packets since node 4 is closer to the *Sink* node; its EDC is lower. On the other hand, in the incorrect case, figure 4.8b, a node that is further from the sink, node 3, acknowledged a packet from a node that is closer to the sink, node 4. This anomaly could be explained by two phenomena. First of them, and the more natural one, is EDC fluctuation, which was presented in section 3.1. This problem can lead to a situation where nodes that actually do not provide routing progress are included in the forwarder set of a given node. In our case with the given topology, it means that an arbitrary forwarder node would have two nodes in its forwarder set. This problem is only present in the ORW, ORWE-BF and ORWE-DC protocols due the fact that the ORWE and ORWE-EDC protocols prohibit any sort of EDC modification during bulk transfer. This is reflected by figure 4.6a where both groups of protocols have similar drops in their reliability just the former group's is more drastic. The second phenomenon resides in the Cooja simulator and stems from its excessive delays in acknowledgement depicted by the two figures in 4.9. The colors in the figures bear with the following meanings:

Green: receiving data on the radio

Blue: transmitting data on the radio

Red: deciding whether to send acknowledgement

Figure 4.9a shows the timeline of a simple acknowledgement, where there is only one node, node 3, sending a packet, and two nodes, node 2 and node 4, are receiving. The decision process for acknowledgement is delayed, but it does not affect the sequence number of the acknowledgement; node 4 acknowledges the packet sent by node 3. While figure 4.9b shows the case when two nodes, node 2 and node 4, are sending two different packets relatively at the same time, and one node, node 3, receives them. In this case, the process time of the first packet on the receiver node, node 3, might delay the acknowledgement decision until the next packet from the other node arrives. The problem is that the routing progress check was done for the first node, the one that is further from the sink, in our case node 2, allowing the



(a) Reliability: All the designs barely lost any packets.

(b) Transmission time: Approximately 2 seconds increase per every forwarder in the ORW design; the ORWE designs stayed on a static level.

(c) Power consumption: Drastic increase in the ORW design; ORWE designs stayed on a static level.

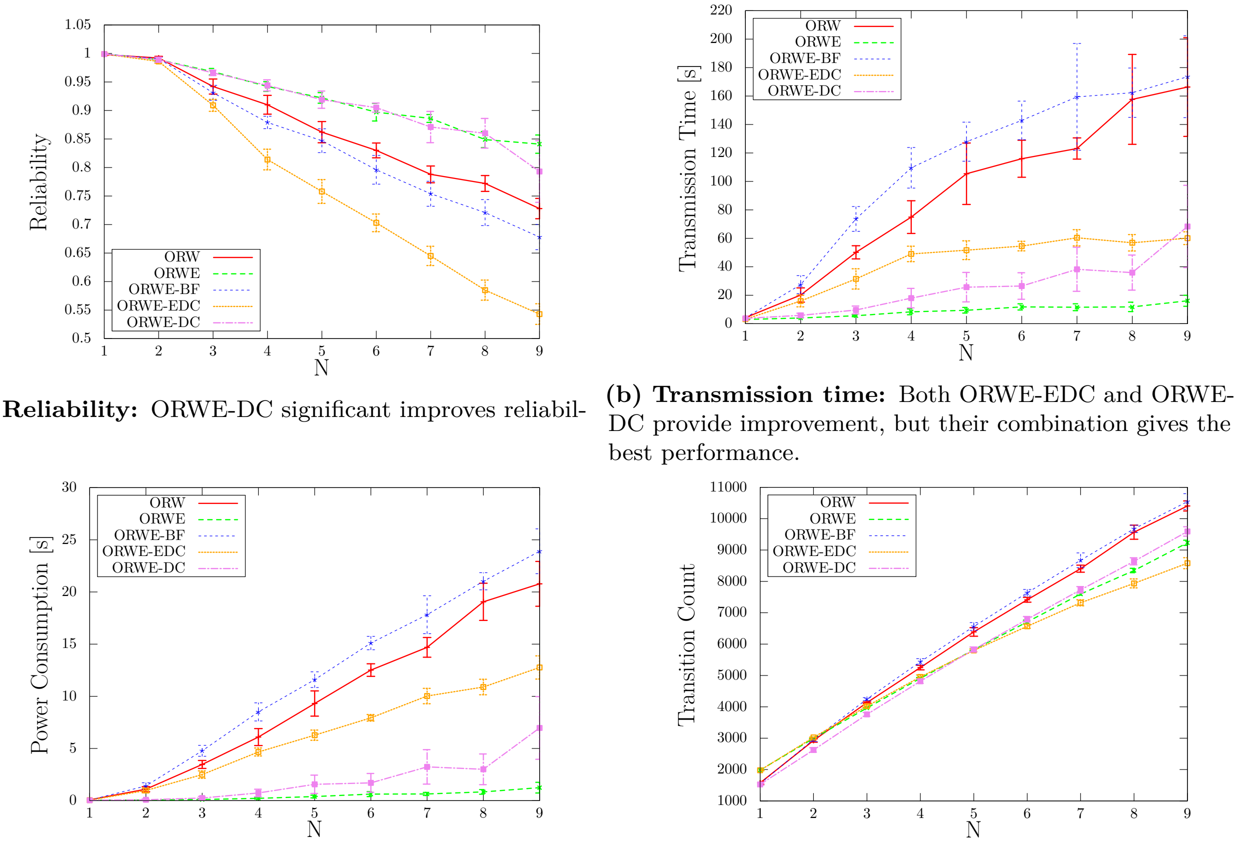
(d) Transition Count: Drastic increase in the ORW design; ORWE-BF and OREW-DC are below the 2000 logical lower bound due to the missing log messages.

Figure 4.5: Core metrics in the function of the number of forwarders using topology from figure 4.4a.

acknowledgement, while the actual acknowledgement sent by the radio transceiver will be sent for the sender of the last received packet. This paradox is verified by figure 4.9b, where node 3 sends the acknowledgement with the sequence number $A9$ instead of $D7$ resulting in a packet loss at node 4 due to the fact that node 4 will assume that another node received its packet. Thus, node 4 will deleted this packet from its message buffer. Meanwhile, it is non-determined which packet is going to be forwarded to the upper layer at the receiver node 3.

A straight-forward workaround for this problem would be to lower the CPU load on each nodes by initiating certain period of gaps between sending two packets on the source node. However, this approach would lessen the meaning of bulk transfer and would truly solve the problem in small-scale topologies where the packet can be transmitted to the sink before the next packet being sent at the source node. If a packet is still in the network when the next packet is being sent on the source, then it is just a matter of time that these packets accumulate on an arbitrary node, probably close to the sink, and overload this node's CPU.

Since the problem itself originates from the fact that a simulation of a hardware component always draws trade-offs in the sense of performance, the ultimate solution for the given problem is to supersede the limited Cooja simulation with tests executed on a real testbed. The next section, section 4.4, shows the evaluation of such measurements for a given large-scale mesh topology involving multiple source nodes and various possible paths between the source nodes and the sink.



(a) Reliability: ORWE-DC significant improves reliability.

(b) Transmission time: Both ORWE-EDC and ORWE-DC provide improvement, but their combination gives the best performance.

(c) Power consumption: Both ORWE-EDC and ORWE-DC provide improvement, but their combination gives the best performance.

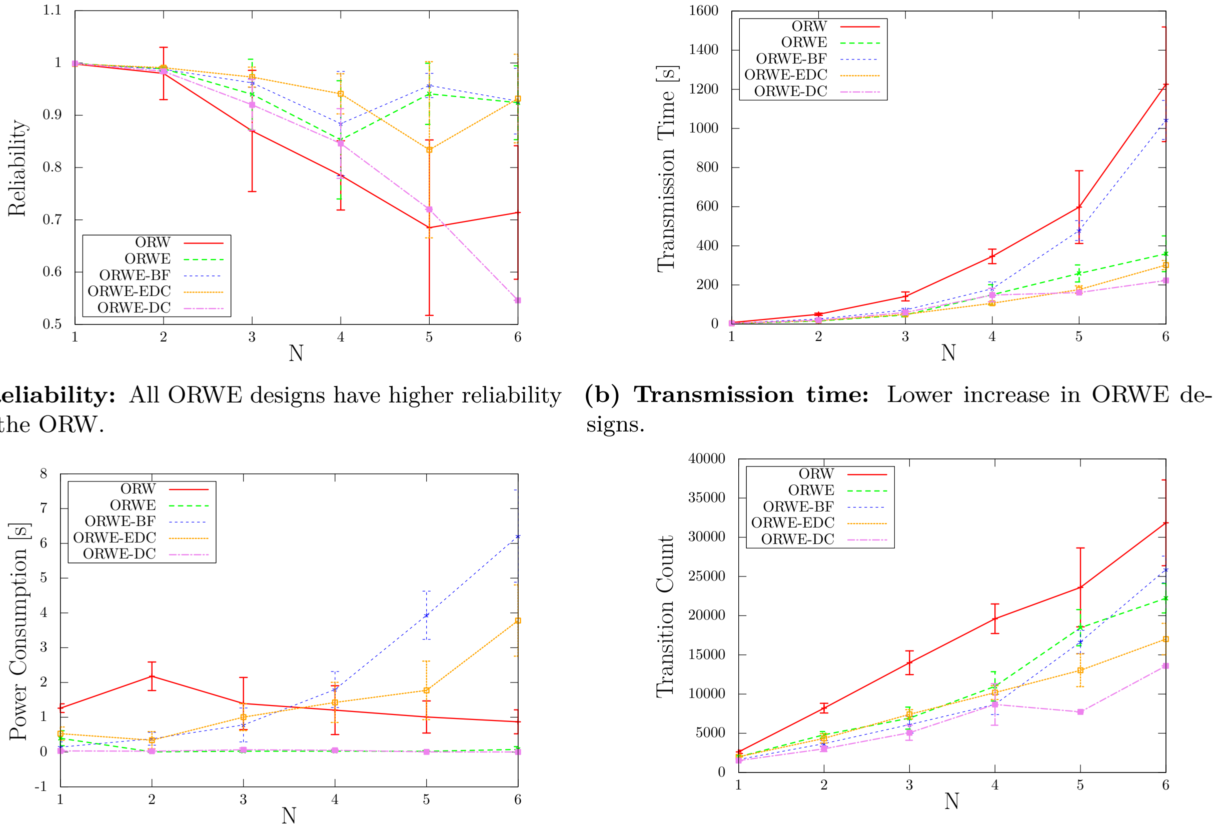
(d) Transition Count: The graphs are similar since there is only one path for all the bulk packets; the only difference stems from the retransmissions.

Figure 4.6: Core metrics in the function of the number of forwarders using topology from figure 4.4b.

4.4 Macrobenchmark

The performance testing in a microscopic scale was in the previous section, section 4.3, using the Cooja simulator. This type of testing provided an insight to the performance improvements provided by the extension protocol. However, as it was pointed out by the end of the section, this type of performance assessment comes with certain limitation on the load that can be put on the simulated nodes. If the load exceeds a certain level, then the measurement gets distorted and eventually becomes inadequate for being a basis of comparisons. Due this limitation, our performance assessment must include another type of evaluation either incorporating a larger scale of network free from the limitation of a simulation. This type of evaluation would be *Indriya*, a publicly available, low-cost, three dimensional wireless sensor network testbed at the National University of Singapore (NUS) using 127 TelosB sensor nodes.

Figure 4.11 shows the connectivity map of the sensor nodes in the Indriya testbed providing a basic understanding of the network structure for the later discussion in this section. There were two sensor nodes dedicated as source nodes, node 11 and 21. Each of these nodes sent 1000 packets as bulk transfer after the end of the configuration phase starting at the same time. The configuration phase involves the activity of all the nodes in the network. When a node wakes up it starts sending packets, with random interval between them, to the sink node. The number of these packets determines the length of the configuration phase. The tests executed



(a) Reliability: All ORWE designs have higher reliability than the ORW.

(b) Transmission time: Lower increase in ORWE designs.

(c) Power consumption: Drastic decrease in ORWE-DC design.

(d) Transition Count: The combination of the ORWE-EDC and the ORWE-DC gives higher transition count than those two designs separately.

Figure 4.7: Core metrics in the function of the number of source nodes using topology from figure 4.4c with $M = 4$.

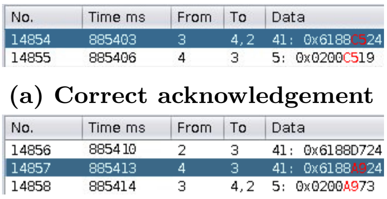
on the Indriya testbed for this project used two packets for the configuration due to the high number of sensor nodes in the network and the limited time resource, 15 minutes, accessible for each test execution.

As for the metrics applied during the evaluation of this benchmark, the macro benchmark follows a slightly different approach than the micro. It does not calculate the average of multiple test runs' results, but rather investigate the features of one specific test execution. Therefore, this approach involves several additional metrics that were not used during the evaluation of the micro benchmark, such as: hop counting, transmission time and rate, average packet delay, and counting both Link and Routing layer duplicates separately, and classify them into two different groups:

1. network wise,
2. source node wise.

The network wise class covers metrics that describe the whole network itself and where the positions of the source nodes are not relevant, while the source node wise class contains metrics where the source node location has effect on the result of the metric. Therefore, mixing those results would corrupt the measured metric data. In case of the hop counts, for instance, the number of hops from the given source node to the sink is highly dependent on the location of this source node in the network.

Table 4.2 shows the results for both metric classes from one execution on the Indriya testbed



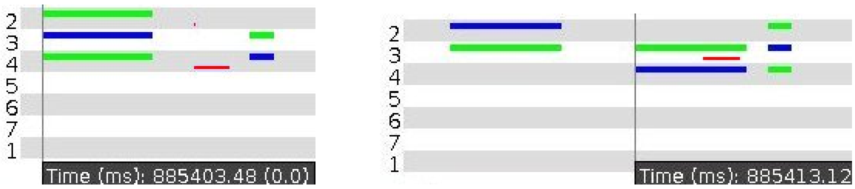
No.	Time ms	From	To	Data
14854	885403	3	4,2	41: 0x6188C524
14855	885406	4	3	5: 0x0200C519

(a) Correct acknowledgement

No.	Time ms	From	To	Data
14856	885410	2	3	41: 0x6188D724
14857	885413	4	3	41: 0x6188A924
14858	885414	3	4,2	5: 0x0200A973

(b) Incorrect acknowledgement

Figure 4.8: Radio Logger: Node 3 with higher EDC acknowledges a packet from node 4 with lower EDC.



(a) Correct acknowledgement: From the two receiver nodes only one acknowledges the packet.

(b) Incorrect acknowledgement: Node 3 checks the EDC for one sender node that gives routing progress but acknowledges the packet from the other sender node that does not give routing progress.

Figure 4.9: Timelines

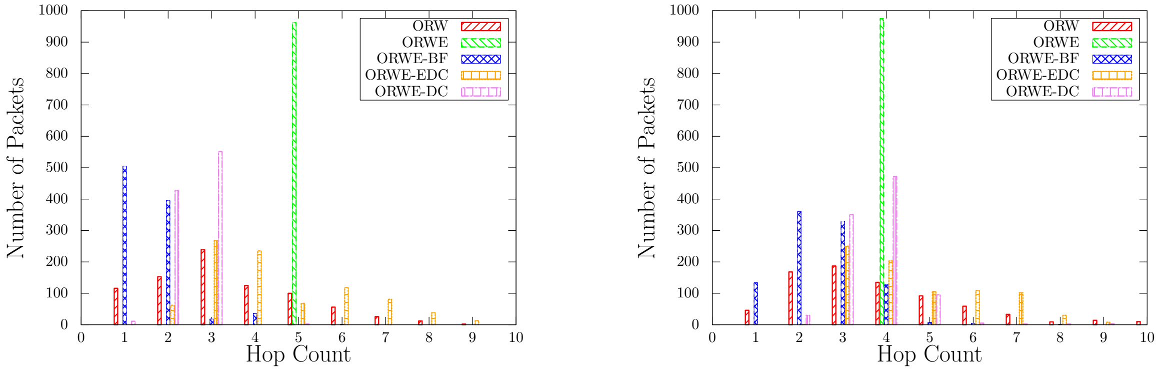
Table 4.2: Performance of different designs on Indriya

Metric	ORW	ORWE	ORWE-BF	ORWE-EDC	ORWE-DC
R	0.954	0.968	0.96	0.863	0.981
DP_{MAC}	15154	1734	1162	3738	1272
DP_{ORW}	6645	33	18	311	253
PC	26.212 s	7.549 s	6.340 s	11.419 s	6.311 s
TX	19564	9501	4404	8954	6926
R_1	0.961	0.975	0.956	0.900	0.991
D_1	4212 ms	716 ms	597 ms	1497 ms	281 ms
HC_1	15.702	5	1.570	4.526	2.550
TT_1	268 s	32 s	109 s	176 s	70 s
TR_1	3.573 p/s	29.168 p/s	9.124 p/s	5.110 p/s	14.100 p/s
R_2	0.944	0.962	0.964	0.827	0.971
D_2	4324 ms	433 ms	1321 ms	3071 ms	502 ms
HC_2	14.120	4	2.514	4.813	3.800
TT_2	228 s	59 s	104 s	129 s	50 s
TR_2	4.124 p/s	16.253 p/s	8.826 p/s	6.407 p/s	19.230 p/s

covering all the five designs. Surprisingly, the best performance was not produced by ORWE, but by ORWE-DC, in the majority of the metrics. One of the main reasons for this surprise can be traced back to the mediocre performance of the ORWE-EDC design in most of the core metrics, e.g., reliability and transmission time. Since both ORWE-EDC and ORWE-DC contain the design solution of ORWE-BF extending it with their own additional solution to another problem not attempted to be solved by ORWE-BF, it can be concluded that the extra solution provided by ORWE-EDC does more harm than good compared to ORW. Therefore, the ORWE design containing the solution of ORWE-EDC produces a lower performance on the Indriya testbed than ORWE-DC, which is free from ORWE-EDC's extra solution.

The second prominent difference between the old and new designs reflected in the two duplicate count metrics. As it was expected, the number of duplicate packets in the network was cut down in the whole network by the initiation of the busy-flag solution. Since this solution is present in the ORWE-BF design, all the other three ORWE designs have similarly low duplicate count. However, ORWE-EDC has approximately three times more Link layer duplicates then ORWE-DC. As a last note on the duplicate count metrics, it is worth noticing that the ratio between Routing layer and Link layer duplicates, however, is relatively high in ORWE-DC compared to the other ORWE designs; ORWE-DC has a roughly 1 to 5 ratio, while ORWE, for instance, has 1 to 52.

As for the power consumption, once again, the ORWE-EDC design seems to degenerate the



(a) Source1

(b) Source2

Figure 4.10: Number of packets with hop count: ORW, ORWE-BF, ORWE-EDC and ORWE-DC designs' hop counts are below the logical lower bound due to missing log messages.

performance of ORWE-BF resulting in an increased power consumption in the final ORWE design compared to either ORWE-BF or ORWE-DC. Comparing the design with the lowest power consumption, ORWE-DC, to the base ORW protocol, it can be seen that more than four times less power was consumed during the use of ORWE-DC than at the use of ORW.

The last network type metric that was not discussed yet is the transition count. In the transition count metric the ORW design uses at least twice as many transitions as any of the ORWE designs do. The simple ORWE-BF design, for instance, performs the two bulk transfers in the Indriya network with almost five times less transitions.

Given that the two source nodes and the sink are marked on the testbed's connectivity map, figure 4.11, the optimal hop count necessary for a packet to get from a source to the sink with the minimal number of hops can be calculated. The optimal hop count for source one is 5 and for source two is 4. After seeing that the average hop count of ORWE-BF, OWRE-EDC and ORWE-DC designs are below this lower bound, the first conclusion from the two seems to be formulated. The simple explanation for the too optimal hop counts is the missing of log messages.

Figure 4.12 and 4.13 show the network traffic sent from each source node for all the designs separately including the Forwarder layer duplicates and packets routed not to the optimal path with the two configuration packets sent before the bulk transfer. The designs that incorporate the busy-flag solution inflict significantly less traffic to the network by choosing one path, probably the optimal, and sending all the bulk packets on that one chosen path. In the figures each node is connected with a line if there was at least one packet transmitted between them. The number placed on the line indicate the number of packets that were transmitted between the two nodes that the line is connecting. Since the positioning of the nodes in the figures are based on the number of transmitted packets between the given two nodes, it might happen in the case of high number of transmitted packets on the same link that the nodes are placed so close to each other that the connecting lines cannot be seen between them. One of the figures that shows this case is 4.12d. From the two sets of figures, it can be seen that the least number of extra hops was produced by the ORWE design. This statement is supported by the two bar charts from 4.10, where ORWE has the most prominent bar with the highest number of packets with the same hop count. For the rest of the designs, there are several bars that below the optimal hop count number. This contradiction, once again, is originated from the deficiency of the logging system that leads to missing log messages.

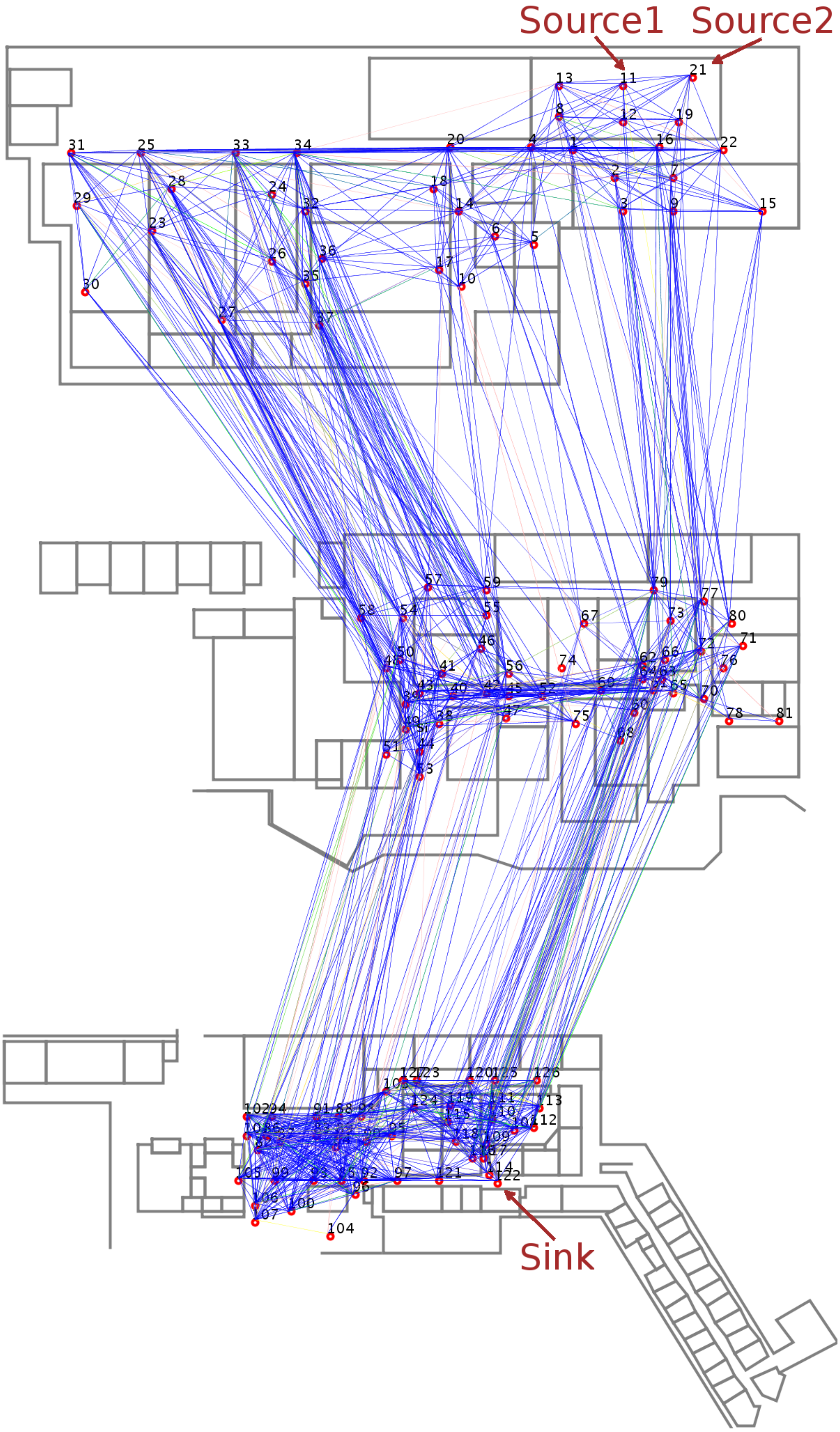
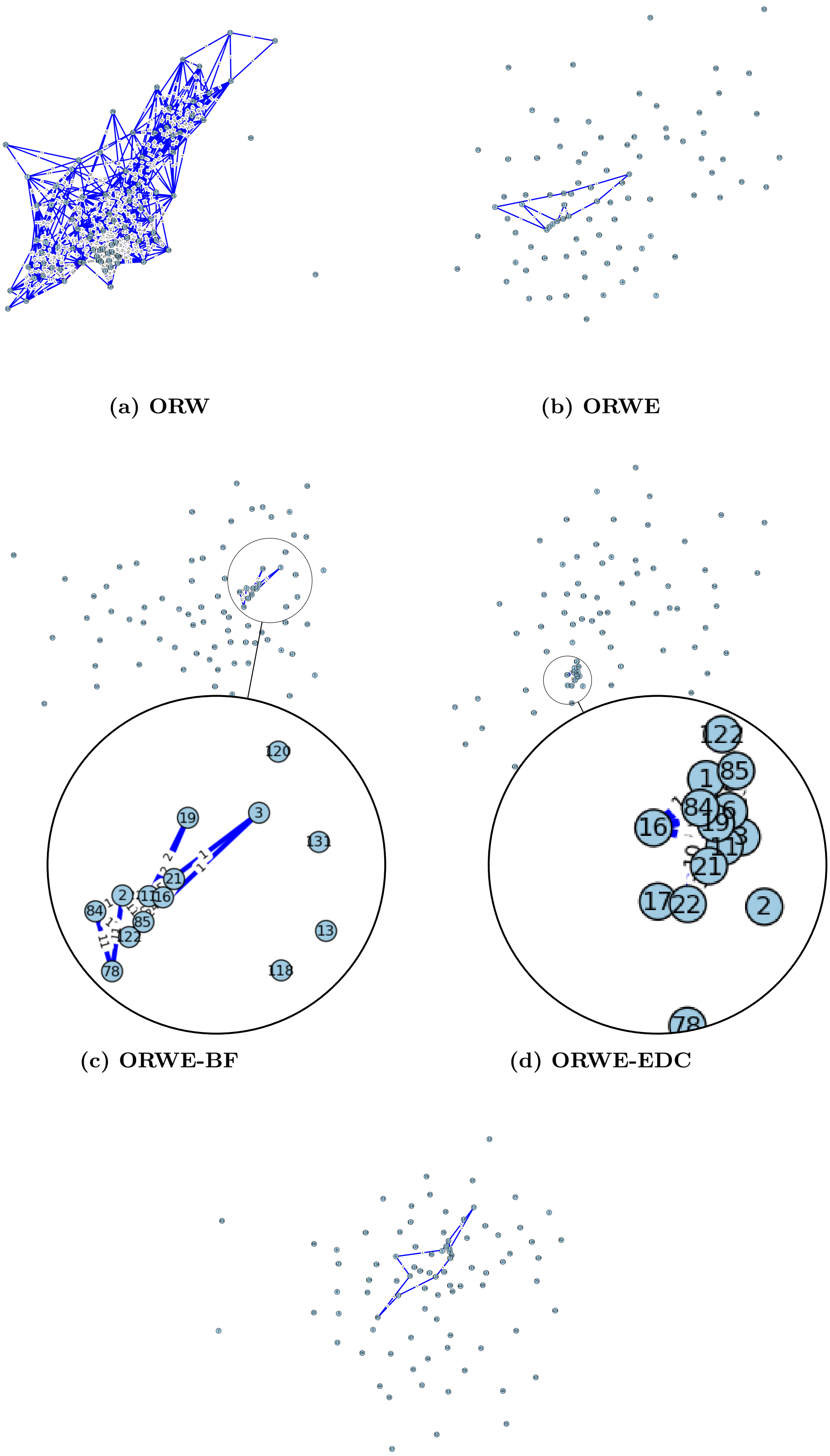


Figure 4.11: Node connectivity over the three floors of Indriya testbed. Taken from [DCA11]



(a) ORW

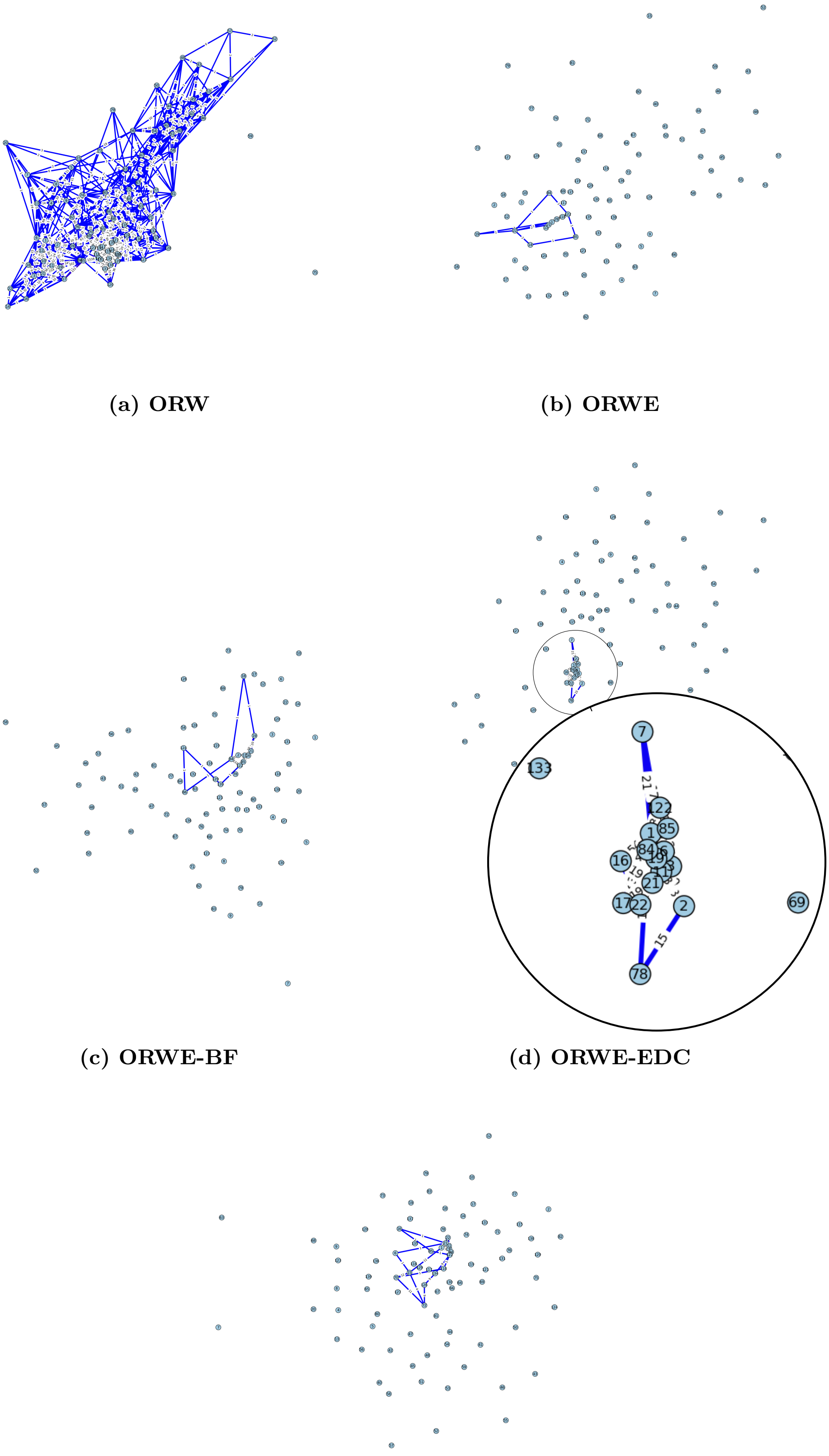
(b) ORWE

(c) ORWE-BF

(d) ORWE-EDC

(e) ORWE-DC

Figure 4.12: Bulk transfer from Source1



(a) **ORW**

(b) **ORWE**

(c) **ORWE-BF**

(d) **ORWE-EDC**

(e) **ORWE-DC**

Figure 4.13: Bulk transfer from Source2

5 Conclusion

First of all, this chapter gives a brief summary about the project covering all the topics that were discussed in the previous chapters. Secondly, there is still place for improvements on the protocol, which are discussed in section 5.2. Lastly, this chapter ends with a final conclusion on this project.

5.1 Summary

The summarization of this project covers a recap of the base protocol including the problems residing in it along with their solutions in the extensions, and a summary about the performance improvements provided by these extensions.

This project was dealing with three main problems of the base protocol, namely ORW, that emerged during a bulk transfer. These problems were the followings:

- collisions,
- EDC fluctuation, and
- early termination of duty cycles during a bulk transfer.

Collisions in the base protocol during bulk transfer can have more serious effects on the overall performance of a network than in a non-bulk packet transfer. The reason for this problem is that the base protocol resolves the colliding acknowledgements on a packet wise basis. In a collision prone situation, such as a bulk transfer on a multiple parallel forwarders topology, the lack of a collision avoidance method leads to a significant increase in the transmission time of the bulk transfer and the power consumption of the whole network, even though the collision detection was working properly avoiding any sort of degradation in the reliability whatsoever. Therefore, such a collision avoidance method was initiated in the extension protocol, and proven to be a valid solution for this problem. While the base protocol had a constant increase in its transmission time, power consumption and transition count by the increasing number of parallel forwarders, the solution from the extensions managed to overcome this increase and keep those three metrics in a static level. The reason for this behaviour can be traced back to the very nature of busy-flagging in the extensions. If a packet is being sent in a bulk transfer and multiple forwarder nodes acknowledge it, then the forwarder nodes have to only resolve the collision for the first packet of the bulk transfer, and the winning node will forward the rest of the transfer; all the other forwarders, in case of no other packets to receiver or sent, will end their duty cycle. Since by this strategy there is no difference between having one forwarder and having more than one forward for a node sending bulk transfer, the power consumption and transmission time do not depend on the number of parallel forwarders. However, the situation where the number of nodes sending bulk transfers exceeds the number of forwarders leads to a case where one node has to receive packets from two source nodes at the same time. This case can result in a moderate degradation of the overall performance of the network. This performance degradation is less drastic than the one that stems from the base protocol's lack of collision avoidance, since there are no extra retransmissions triggered due to collision. The only problem is that the load is not balanced among all the forwarder nodes, but incurred on a subset of them only, where the size of this subset is tantamount to the difference between the number of forwarders and the number of nodes sending a bulk transfer.

The EDC fluctuation is the phenomenon that allows packets to traverse back in their path, and thus being falsely identified as duplicate packets eventually resulting in the loss of those packets. One of the extensions provides a fairly simple solution for this problem by forbidding EDC updates during a bulk transfer for the nodes that are involved in that given bulk transfer. This approach mitigates the effects of this problem inflicted on the metrics of power consumption and transmission time. For instance, at the topology where the six forwarder nodes are placed in a serial line between the source of a bulk transfer and the sink node, this solution needed half of the time for performing a bulk transfer than the time needed by using the base protocol.

The merit of duty cycling can turn into a defect when it used at the wrong time. If a node ends its duty cycle prematurely, then it can lead to an increase in transmission time, since the node that trying to send packets to this node has to wait until the node turn on its radio receiver again. The solution for this problem is similar to the solution that was applied for solving the EDC fluctuation problem. In more details, it will not allow to a node to end its duty cycle if that node is either receiving bulk transfer packets or sending them. By this approach a node does not terminate its duty cycle until the transmission of the bulk transfer has not ended. At the same topology that was used for the example at the EDC fluctuation problem, the topology with the six forwarder nodes placed in a serial line between the source and sink nodes, this approach produced even better results than the solution for the EDC fluctuation. A bulk transfer was performed three times faster when the termination of the duty cycle was forbidden during a bulk transfer than in the base protocol.

5.2 Future Work

This section provides a list of suggestions for future improvements to the set of extensions presented in this paper.

More Evaluation on Real Testbeds: One of the several possibilities for future work would be to evaluate the performance of the extensions on several other real testbeds, such as: Motelab [WSW05], Kansei [Ert+06], Twist [Han+], etc. The structural diversity of these networks would give a remarkable opportunity for locating latent anomalies remained hidden from the tests executed on the Indriya testbed waiting to be revealed by future wireless sensor network enthusiasts.

Evaluation of Lossy Nodes Scenarios: Another feasible extension for the evaluation would be to test the protocol with lossy nodes having low link quality. Such testing would allow the protocol to be applicable in additional areas, e.g., volcanoes.

Refactoring of the Logging System: Throughout the evaluation of this project, lavishing inconsistencies revealed themselves between the sheer results and their logical bounds. Struggling to conceive this copious discrepancy, an eminent WSN zealot being involved in this project might consider the refactoring of the well shunned logging systems, the sole soul of the vanishing logs anomaly.

5.3 Conclusion

This project showed that the low power, low delay ORW routing protocol is capable of handling multiple, concurrent, bulk transfers with high-throughput by applying the three extensions described in this paper without suffering any sort of performance degradation in metrics such as power consumption, transition count and reliability. Conversely, by applying these extensions a significant performance improvement can be obtained in all the metrics that were presented in this report. Lastly, these extensions have no effect on the non-bulk packet transfer, since they are only activated when the transition rate exceeds a certain level; thus providing backward compatibility for the base ORW protocol.

In conclusion, all the performance improvements provided by the extensions are based on one capability. The very same capability that allows us, mortal people, to perform our common, interference free, daily conversations. It is the capability to *remember*. By remembering we can minimize the chance of interference among multiple concurrent conversations similarly as the extensions of the ORW protocol could lower the frequency of colliding packets during a bulk transfer. The essence of such a policy in a routing protocol can be well expressed by a quotation from George Santayana: ”Those who cannot remember the past are condemned to repeat it”[San05, p. 284].

References

[adv13] advanticsys. *802.15.4 TelosB mote Module including temperature, humidity and light sensor.* 26 July, 2013. URL: `http://www.advanticsys.com/shop/mtmcm5000msp-p-14.html`.

[BM05] S. Biswas and R. Morris. ExOR: Opportunistic Multi-Hop Routing for Wireless Networks. *SIGCOMM '05 Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications* (2005). DOI: `10.1145/1080091.1080108`.

[Cha+07] S. Chachulski et al. Trading structure for randomness in wireless opportunistic routing. *ACM SIGCOMM Computer Communication Review* **37**.4 (Oct. 2007). DOI: `10.1145/1282380.1282400`.

[DCA11] M. Doddavenkatappa, M. Chan, and A. Ananda. Indriya: A Low-Cost, 3D Wireless Sensor Network Testbed (2011).

[DÖD11] S. Duquennoy, F. Österlind, and A. Dunkels. Lossy Links, Low Power, High Throughput (2011).

[DP10] W. Dargie and C. Poellabauer. *Fundementals of Wireless Sensor Networks.* John Wiley & Sons Ltd., 2010. ISBN: 978-0-470-99765-9.

[Ert+06] E. Ertin et al. A Testbed for Sensing at Scale (2006).

[Han+] V. Handziski et al. TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks ().

[Kim+07] S. Kim et al. Flush: A Reliable Bulk Transfer Protocol for Multihop Wireless Network (2007).

[Lan+12] O. Landsiedel et al. Low Power, Low Delay: Opportunistic Routing meets Duty Cycling. *IPSN '12 Proceedings of the 11th international conference on Information Processing in Sensor Networks* (2012). DOI: `10.1145/2185677.2185731`.

[LG09] P. Levis and D. Gay. *TinyOS Programming.* Cambridge University Press, Apr. 2009. ISBN: 978-0521896061.

[Ram+10] B. Raman et al. PIP: A Connection-Oriented, Multi-Hop, Mutli-Channel TDMA-based MAC for High Throughput Bulk Transfer (2010).

[San05] G. Santayana. *The Life of Reason; or the Phases of Human Progress.* Charles Scribner's Sons, 1905.

[Tan11] O. Tange. GNU Parallel - The Command-Line Power Tool. *;login: The USENIX Magazine* **36**.1 (Feb. 2011), 42–47. URL: `http://www.gnu.org/s/parallel`.

[WSW05] G. Werner-Allen, P. Swieskowski, and M. Welsh. MoteLab: A Wireless Sensor Network Testbed (2005).