Forwarding Strategies for 6LoWPAN-Fragmented IPv6 Datagrams

Vom Promotionsausschuss der
Technischen Universität Hamburg-Harburg

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation

von
Andreas Weigel

aus
Potsdam, Deutschland

2017

Date of Oral Examination	September 05^{th}, 2017
Chair of Examination Board	Prof. Dr. Heiko Falk Institute of Embedded Systems Hamburg University of Technology
First Examiner	Prof. Dr. Volker Turau Institute of Telematics Hamburg University of Technology
Second Examiner	Prof. Dr. Andreas Timm-Giel Institute of Communication Networks Hamburg University of Technology

Acknowledgment

Several people supported me in the long, difficult and sometimes frustrating process of finishing this dissertation. I want to seize the opportunity to express my deeply felt gratitude towards them.

First, I would like to thank my supervisor Prof. Turau for his guidance, encouragement and intellectual input, but also for being the kind of superior he is.

I would like to thank my colleagues, who made everydays work at the institute a pleasant experience. Special thanks go to my "roommates" Bernd-Christian Renner, Martin Ringwelski and Florian Kauer for bearing with me and my curses and complains and for providing so much valuable input. Further, I'd like to thank Stefan Unterschütz and Martin Ringwelski for their implementation work on CometOS and its 6LoWPAN module and the numerous fruitful discussions.

I want to thank my parents for their genes and the continuous support I experienced throughout my life. And finally: Thanks Susanne, for your encouragement and help and for taking on life together with me.

Andreas Weigel
Lüneburg, November 2017

Abstract

Recent efforts towards a fully standardized protocol stack (RPL, CoAP, 6LoWPAN) for "low power and lossy networks" (LLNs) contribute to realize the vision of the Internet of Things. 6LoWPAN is a central building block among these protocols, enabling the transmission of IPv6 datagrams using the IEEE 802.15.4 protocol for wireless mesh networks. To provide IPv6' minimal MTU of 1280 bytes by means of the 127 byte frames of IEEE 802.15.4, 6LoWPAN defines a fragmentation mechanism. However, the use of fragmentation is suspected to amplify existing problems for the communication in LLNs and thereby decrease reliability.

This dissertation explores the impact of 6LoWPAN fragmentation on the reliability of transmissions for the LLN-typical collection traffic pattern and implementation strategies to improve this reliability. The two route-over forwarding methods "Assembly" and "Direct" are considered as basic implementations. The former reassembles a datagram at every IPv6 hop, the latter uses a cross-layer approach to forward individual fragments as soon as they arrive.

An extension to a bit-error-based analytical model for fragmented 6LoWPAN transmissions is developed to estimate the number of created frames and to better reflect current implementation realities. Furthermore, simulative and testbed studies are carried out to evaluate the basic implementations and the extension developed as part of the dissertation, adaptive rate-restriction.

Based on the results of these parameter studies, the author proposes a mechanism called "6LoWPAN Ordered Forwarding" (6LoOF), which is designed to prevent local short-term congestion by suspending transmissions at nodes that overhear ongoing transmissions in the neighborhood. Evaluation of 6LoOf in simulation and two different testbeds show that – especially in combination with adaptive rate restriction – it significantly improves the reliability of the transmission for large 6LoWPAN-fragmented IPv6 datagrams, in some cases reducing the number of dropped datagrams by 50 %.

Furthermore, inconsistent results between simulation and testbed triggered a detailed analysis of the used MAC layer, which relied on the CSMA/CA mechanism provided by the transceiver hardware. This experimental analysis shows that the realization of IEEE 802.15.4 in hardware on many current transceivers severely impacts the reliability of transmissions in multi-hop traffic scenarios, potentially biasing results of research obtained with communication stacks using these realizations.

Contents

List of Figures

List of Tables

1 Introduction

Since the beginning of the 2000s, a class of networks termed wireless sensor networks (WSNs) have received the attention of a large research community. These usually feature a large number of small, resource- and energy-constrained devices that form a wireless mesh network to realize a sensing, monitoring or control task. At the time of writing, a typical node can be expected to possess from 4 to 32 KiB of RAM and 64 to 512 KiB of program memory, drawing current in the order of a few tens of mA with the transceiver active and a few µA when it is sleeping. Due to the nature of wireless communication channels, links between devices are often asymmetric and lossy, prone to interference by other wireless technologies and of transient nature due to changes in the environment. These properties also lead to the classification of low power, lossy networks (LLNs) for typical WSNs.

In recent years new names like cyber-physical systems, internet of things or industry 4.0 have emerged and show that the interest in ubiquitous autonomously communicating systems is unbroken.

Said attention brought forth a large number of protocols specifically tailored to cater the specialties of WSNs. Ranging from the "alphabet soup" of MAC protocols ([Ali+06]) over a plethora of routing protocols and corresponding link-quality metrics to transport protocols replacing the ubiquitous but for wireless lossy communication not terribly well-suited TCP, all layers of the communication stack have received due attention. When the dust settled, standardization efforts were launched to order the chaos.

Industry standards like ZigBee and WirelessHART based on the IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (IEEE 802.15.4) were among the first of such efforts. Some years later, the Internet Engineering Task Force (IETF) instituted several working groups dealing with standardization of protocols for LLNs. Among them, the "IPv6 over Networks of Resource-Constrained Nodes" (6lo) defined mechanisms to enable the transmission of IPv6 datagrams over IEEE 802.15.4 networks, called 6LoWPAN. Its main responsibilities are compression of the comparatively large IPv6 and UDP/TCP headers to prevent the huge control overhead in combination with 127 B payload in standard IEEE 802.15.4 frames and fragmentation of large datagrams that do not fit a IEEE 802.15.4 frame even after compression. The Routing Protocol for Low Power and Lossy Networks (RPL) and the Constrained Application Protocol (CoAP) complete the fully standardized stack for that class of networks.

Considering the fragmentation of large datagrams it is intuitively clear that splitting up a datagram and transmitting the individual fragments one after the other does not improve the overall reliability of the reception of a datagram. Every single fragment has to arrive for the datagram to be successfully received and on paths that incorporate several wireless transmission hops, sending out a whole bunch of them can further degrade the reliability when frames belonging to the same datagram content with each other to acquire the wireless channel, that is, if they they can even "hear" each

other – otherwise, senders along the same path are likely to cause hidden terminal collisions between consecutive fragment transmissions.

While arguably a large number of applications can be satisfied with small data payloads and low data rates and therefore are not overly concerned the issue of fragmentation, a number of applications with demand for large payloads and data rates exist. Examples for such applications are smart metering and structural health monitoring. Both produce comparatively large application data that periodically has to be collected and forwarded or processed. In the other traffic direction, over-the-air-programming (OTAP) of nodes usually is concerned with the transport of large data blobs to reprogram nodes within a wireless network.

With fragmentation being expected to have some impact on the performance of transmissions of large datagrams, it is desirable to have some quantitative information available on exactly how strong this impact can be. At the time of writing, several studies exist that examine the performance of 6LoWPAN fragmentation using either analytical models or in most cases very simple experimental setups with only a few number of wireless hops. All of them only cover the most basic forwarding strategies. While some problems are identified, at the moment no comprehensive evaluation of the 6LoWPAN fragmentation in more realistic multi-hop network environments and considering enhancements to the forwarding exist.

This dissertation aims at providing such a comprehensive overview over 6LoWPAN fragmentation and contains several contributions towards this aim. To be able to better assess the influence of fragmentation on reliability, an extension to an existing analytical model that better captures the realities of current 6LoWPAN implementations is presented. With the help of the model, it is possible to get an estimate of the impact of fragmentation in multi-hop networks.

Furthermore, a parameter study with regard to the IEEE 802.15.4 MAC and implementation parameters like the available data buffer size is carried out in simulation and a testbed of 13 nodes. It provides an overview about suitable configuration of the underlying IEEE 802.15.4 MAC in multi-hop traffic collection scenarios.

Because of initially inexplicable results in various testbed setups that deviated strongly from corresponding simulations, a detailed examination of the state of the MAC and PHY layers was carried out and revealed that the implementation of the so-called “extended operating mode” of the used transceiver hardware caused the reliability of transmissions to drop dramatically. While this effect is especially strong for the traffic pattern caused by 6LoWPAN fragmentation, it can be generalized to other scenarios as well and may cause bias to experiment results, whenever the extended operating mode of this transceiver or similar modes of operation on other transceivers is used.

To improve the overall reliability of fragmented transmissions, a novel forwarding strategy is proposed. The 6LoWPAN ordered forwarding (6LoOF) protocol is designed to reduce contention for the wireless channel between nodes especially in collection traffic scenarios, while being compliant to the 6LoWPAN standard. A thorough evaluation of 6LoOF is presented in simulation and two testbed scenarios, facilitating the open experiment platform of the IoTLab.

Some of the above mentioned contributions have also been published as a research paper or article. Some chapters of this dissertation are based on and reuse parts of

these papers. The following list provides an overview on the publications reappearing in this dissertation and clarifies the part of work done by me and the other authors.

- Chapter 3 is based on [WT14], which was created by me.
- Chapter 5 is based on [Wei+14b]. Martin Ringwelski provided the majority of the 6LoWPAN implementation for CometOS, the idea to the progress-based retry control (PRC) forwarding mode and was involved in the evaluation. I developed the Direct-ARR mode, created most simulation scenarios and was responsible for a major part of the evaluation. Andreas Timm-Giel and Volker Turau gave feedback and made suggestions with regard to evaluation and editing.
- Chapter 6 is based on [WT15], which was created by me.
- Chapter 4 introduces CometOS ([UWT12]), which was initiated by Stefan Unterschütz and developed by Stefan Unterschütz, me and Florian Kauer.

This dissertation is structured as follows: Chapter 2 introduces the problem domain, points out the most important protocols and approaches and defines the research goals of the dissertation. The analytical model developed as part of this dissertation is introduced in Chapter 3. This chapter also discusses the output of the model for a certain sets of inputs, including different paths lengths, number of fragments, retransmissions and different forwarding strategies. Chapter 4 describes the used frameworks and tools for simulation environment and testbed deployments and the simulation model. Furthermore, the approach to derive simulation models from testbed deployments that serves as a validation mechanism of the simulation model is introduced. Chapter 5 contains a study on basic forwarding strategies for 6LoWPAN fragments of large IPv6 datagrams. Due to a combination of sub-optimal physical link layer model and the transceiver's operating mode chosen for the testbed, this chapter can be characterized as a "lessons learned" chapter. The issue of this operating mode is examined in detail in Chapter 6. An experimental methodology to assess the impact of the used transceivers "extended operating mode" is developed and applied. Chapter 7 contains a revised parameter study of 6LoWPAN forwarding strategies using simulations and a testbed environment. The 6LoOF protocol is introduced and described in detail in Chapter 8. Furthermore, an evaluation of the 6LoOF protocol in comparison to the basic forwarding modes is presented. The dissertation is concluded in chapter 9.

2 Problem Statement

This chapter introduces IEEE 802.15.4 and 6LoWPAN, discusses basic forwarding strategies for 6LoWPAN fragmentation, and introduces a typical protocol stack for the Internet of Things (IoT). Application scenarios are presented to support the significance of evaluating 6LoWPAN fragmentation performance. Furthermore, the operation in energy-constrained networks is discussed and goals of the experimental and simulative evaluation carried out in this dissertation are stated.

In this and the following chapters, the text refers to units of data that are transmitted by a node or a protocol layer, i.e., "packets". To avoid confusion, in this dissertation the following nomenclature is used:

- **frame**: A data frame (header + PHY service data unit (PSDU)) in context of the IEEE 802.15.4 protocol, i.e., data packets used by the link layer.
- **fragment**: A frame carrying 6LoWPAN fragmentation information and part of a datagram as payload.
- **datagram**: An IPv6 datagram. Represented by multiple fragments, if 6LoWPAN fragmentation is applied.
- **packet**: A datagram carrying UDP header and payload of the TCP/IP application layer. In the context of this dissertation it translates into a single IPv6 datagram.

2.1 IEEE 802.15.4

The "IEEE Standard for Local and metropolitan area networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)" (in this thesis referred to as IEEE 802.15.4) was first published in 2003, with major revisions in 2006, 2011 and 2016 [06; 11a; 16]. It defines several physical (PHY) layers and a medium-access layer (with several extensions) for low-rate wireless networks. One of the most widely used PHYs for sensing application, for which also a large number of transceivers is available, is the one operating in the 2.4 GHz ISM band. IEEE 802.15.4 is used as PHY and MAC layer for the industry standard ZigBee [12b] and the IETF standard 6LoWPAN [Mon+07].

IEEE 802.15.4 defines two general operating modes: beacon-enabled and non-beacon-enabled. The former employs so-called beacons, which are regularly broadcasted by the coordinator of a personal area network (PAN coordinator). By means of those beacons, a superframe structure is established, which consists of a contention access period (CAP) and a contention-free period (CFP). In the former, nodes content for access of the channel using a slotted carrier sense multiple access with collision avoidance (CSMA/CA) protocol and may also try to allocate a guaranteed time slot

(GTS) from the CFP. In the latter, nodes can use a previously allocated GTS to communicate with the PAN coordinator. During the guaranteed time slots that are not allocated to a node, this node may turn off its transceiver to save energy, which is the only possibility for duty cycling explicitly defined by the standard. Other protocols may define low-power listening or low-power probing techniques but are out of the scope of the IEEE 802.15.4 standard.

Until recently, the beacon-enabled mode only supported single hop, i.e., star topologies. An extension to IEEE 802.15.4, the distributed synchronous multi-channel extension to IEEE 802.15.4 (DSME) [12a], describes a method to extend this TDMA scheme to multiple hops and multiple channels. A similar direction takes another extension named TSCH [WPG15], which also uses a TDMA scheme, albeit without making use of the beacon-enabled mode's superframe structure. TSCH is derived from the industry standard WirelessHART [10]. While TSCH defines the mechanisms for nodes to communicate according to an existing communication schedule, it does not provide any protocols to actually establish such a schedule. Recent research efforts in that direction are Orchestra [Duq+15] and 6top, which is a standardization effort by the IETF currently in draft state [WV16].

The non-beacon mode operates without any superframe structure or regular beacons. Nodes transmit frames using an unslotted CSMA/CA protocol, which includes a random backoff period, a clear channel assessment (CCA) and subsequent backoffs in case the channel is considered busy.

Independently of the mode used, IEEE 802.15.4 defines retransmissions and acknowledgment frames for unicast transmissions. Receivers of a frame transmit an acknowledgment after they receive a unicast frame with the `Ack Request` flag set in the frame control field. The ACK is sent after a short delay without executing the CSMA/CA mechanism.

2.2 6LoWPAN

Being the protocol this thesis examines, the 6LoWPAN protocol is introduced in this section, with the main focus on 6LoWPAN fragmentation.

2.2.1 Compression and Fragmentation

The reasons for the existence of the 6LoWPAN protocol are twofold: First, IPv6 specifies a minimal maximum transmission unit (MTU) of 1280 B for any link-layer protocol below it. To transport IPv6 datagrams over an IEEE 802.15.4 link-layer with a maximum PHY layer payload of 127 B, fragmentation at the 6LoWPAN layer is necessary to present an interface to the IPv6 layer that supports a sufficiently large MTU.

Secondly, the headers for IPv6 (40 B) and UDP (8 B) or TCP (20 B) are large compared to the typical maximum PHY frame payload of 127 B. The IEEE 802.15.4 MAC header occupies up to 25 B and AES-CCM-128 encryption may use up another 21 B. Hence, the available payload size for a UDP packet in one frame can be reduced to 33 B. Complete use of these 33 B yields an overhead ratio of 74.4 % (adding 2 B for PHY header and the start of frame delimiter (SFD)). To reduce this high overhead ratio, 6LoWPAN defines several compression algorithms (HC1 and HC2), which in

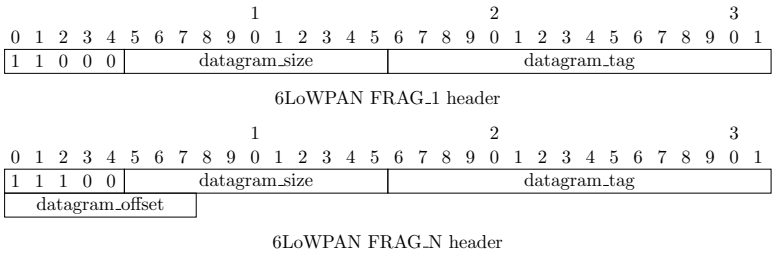


Figure 2.1: 6LoWPAN fragmentation headers

turn are updated by 6LoWPAN IPHC header compression (IPHC) [HT11]. While compression is an important topic especially for small IPv6 datagrams with only a handful of bytes payload, it is seen as a problem orthogonal to the performance of the fragmentation mechanism.

To implement fragmentation, 6LoWPAN defines two different fragmentation headers, one for a first fragment (FRAG_1) and a different one for any subsequent fragment (FRAG_N; Fig. 2.1). Both include the uncompressed size of the IPv6 datagram and a tag to identify the datagram the fragment belongs to. The FRAG_N header additionally carries an offset field, which defines the position of the fragment within the whole datagram given in a unit of 8 octets.

Provided with experience concerning the fragmentation of large data blocks while working at the iEZMesh project, I expected fragmentation to amplify existing problems in multi-hop wireless mesh networks. iEZMesh was a project funded by the German government. One of the application requirements identified for the project was the collection of smart meter measurement tables sized 1 kB to 3 kB. With the link-layer supporting frame sizes of 128 B, the preconditions are similar to those found with large fragmented IPv6 datagrams over IEEE 802.15.4. To satisfy the requirements, we implemented a fragmentation mechanism at the transport layer, together with an actual retransmission scheme for individual fragments based on negative acknowledgments. The evaluation of the performance yielded significant reliability issues for the large data blocks [Wei+14a], even in the presence of the mechanism for retransmissions.

Considering that the loss of a single fragment leads to the loss of a whole datagram and the fact that typical wireless transmissions are inherently lossy due to the properties of wireless channels, collisions and interference, I expect that 6LoWPAN fragmentation is confronted with similar issues and that transmissions of large IPv6 datagrams via 6LoWPAN may exhibit low reliability. Further it is to be expected that the impact of fragmentation increases with the number of fragments and the length of a route. These considerations motivate the evaluation of the performance of fragmentation and several forwarding strategies and the development of the new forwarding protocol 6LoOF, which are presented in this thesis.

2.2.2 6LoWPAN Routing Schemes

With 6LoWPAN, routing in general can be performed at two different layers. First, a layer at the level 2.5 of the 6LoWPAN adaptation layer implements the routing. In that case, some mesh routing protocol has to emulate a full broadcast domain at the physical level for the IPv6 layer. This variant is called ımesh-under routing [HC08; Cho+09]. Thereby, link-local addresses and link-local multi-cast can easily be used from an IPv6 layer perspective and IPv6-based protocols can theoretically be left unchanged. An example for this is the neighbor discovery protocol [Nar+07]. Neighbor discovery makes extensive use of link-layer multicasts, which have to be translated to flooding the mesh network. This means that the mesh routing protocol has to provide potentially complex mechanisms to offer reliable operation over a multi-hop mesh network, which is far from trivial. Moreover, such mechanisms are already available at the IPv6 network layer and have to be recreated for the layer 2.5 mesh routing [HC08].

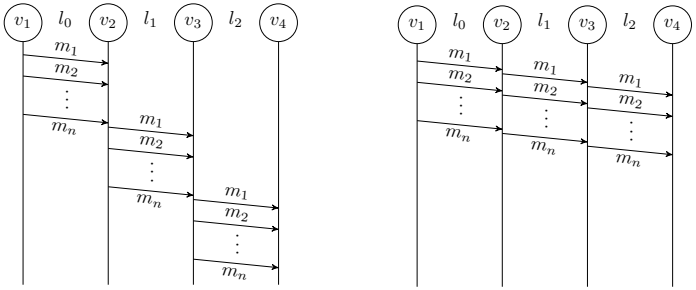
The other possibility is to delegate routing decisions to the IPv6 layer: Every hop in a meshed 6LoWPAN network becomes an IPv6 routing hop. This routing scheme is called ıroute-over [HC08]. Using route-over has several implications. First, global IPv6 addresses ([Nar+07]) have to be used, because IPv6 forbids routing of link-local addresses. That makes the original HC1 and HC2 compression algorithms impractical and is one reason for the introduction of the IPHC and 6LoWPAN next header compression (NHC) [HT11]. With regard to fragmentation, route-over also means that datagrams – sticking to a strict separation of layers – have to be reassembled at each intermediate hop of the 6LoWPAN mesh network, because fragmentation is then handled below the network layer.

With the creation of RPL [Win+12], a standardized routing protocol for low-power and lossy networks at the IPv6 layer is available to be used with a route-over routing scheme, which perfectly fits the usual demands on routing protocols for typical 6LoWPAN wireless mesh networks. Considering the arguments, I decided to focus on the evaluation of route-over, as it allows building a completely standardized protocol stack and avoids the awkward emulation of a single hop broadcast domain above a lossy multi-hop wireless mesh network.

2.2.3 Basic Route-Over Forwarding Techniques

As described in Sect. 2.2.2, using the route-over routing scheme implies reassembling a datagram at every intermediate node of the wireless mesh network. This is the first basic and most straightforward forwarding strategy and is called *Assembly* or *Assembly mode* throughout the thesis. During the whole process of reassembling the datagram, all fragments have to be stored in some buffer, even at intermediate nodes, which are not concerned with the content of the datagram. Hence, for each datagram in transit, buffer space for the whole datagram has to be available. Considering typical resource-constraint hardware for wireless sensor networks, this is a non-negligible issue. Furthermore, reassembling at every intermediate hop prevents pipelining of fragments on longer (> 3) paths. Therefore, an unnecessary large end-to-end latency can be expected (Fig. 2.2).

In contrast to this approach, which strictly preserves layer separation, a cross-layer approach can be employed, which is called *Direct* or *Direct mode* in the remainder of



(a) Assembly (b) Direct

Figure 2.2: Message flow in Assembly and Direct modes. The Direct mode has potential for pipelining as well as an increased probability for collisions.

the thesis. For each incoming first fragment, the information necessary to identify and process subsequent fragments of the datagram is stored in a "virtual fragment buffer". The buffer is called virtual, because it does not store the payload data the fragment carries, but only the metadata, i.e., information about progress and identity of the fragment. The fragment itself is immediately scheduled for transmission to the next hop, which is queried directly from the IPv6 layer. This is always possible, because the IPv6 header does always fit the first fragment. Subsequent fragments then are matched against the entries in the virtual fragment buffer and routed along the same path. Note that the Assembly mode also uses the same path, but the moment at which the routing decision is taken is different: with Assembly, it is the reception of the last fragment, with Direct, the reception of the first fragment.

Using the Direct mode, datagrams have only to be stored for reassembly at their IPv6 destination (or the 6LoWPAN border router). Hence, it provides good potential for saving buffer space. Furthermore, pipelining on long paths becomes possible and thereby the overall latency can potentially be reduced. On the other hand, immediate forwarding also gives rise to self-interference. Fragments are prone to interfere with their predecessors, which have already advanced on the routing path. This can be especially harmful at the node two hops farther down the path, because such a node usually will be a hidden-terminal. In this situation, the clear-channel assessment part of IEEE 802.15.4's CSMA/CA algorithm is not able to prevent a collision. This increased potential for collisions is also implied in Fig. 2.2.

2.2.4 Adjacent Protocols

The IPv6-based standardized protocol stack for resource constrained networks and devices then could be the one shown in Fig. 2.3.

Above the described combination of a route-over 6LoWPAN adaption layer and the IPv6 network layer with RPL as routing protocol, the combination of UDP [Pos80] and CoAP [SHB14] is used at the transport and application layers.

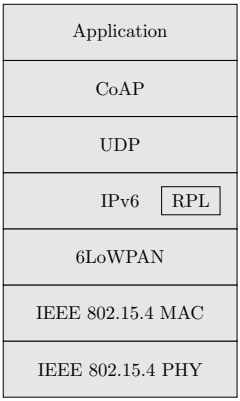


Figure 2.3: Standard protocol stack for low-power lossy networks

CoAP is similar in spirit to Hypertext Transfer Protocol (HTTP) in defining addressable resources as RESTful services. Additionally, it defines simple reliability features, a basic congestion control mechanism and a binary representation to increase efficiency. Hence, it takes on some responsibilities of a transport layer. Current standardization and research efforts for CoAP focus, among others, on congestion control [BGD15; Bet+15; JDK15] and blockwise CoAP transport [BS16]. The latter introduces a mechanism to CoAP to split up large payloads into smaller blocks with the target of avoiding to burden lower layers with "conversation state that is better managed in the application layer"[BS16]. This includes 6LoWPAN fragmentation and aims at reducing the need for it and thereby is fundamentally different from the approach presented in this thesis, which is to improve the performance of 6LoWPAN fragmentation.

The RPL protocol [Win+12] defines a tree-based routing for low-power lossy networks, such as wireless sensor networks or cyber-physical systems. Its main ideas are derived from the collection tree protocol [Gna+09]. It builds bi-directional routing trees by two core mechanisms:

- Beacons called DIOs are used to form routes towards a single destination: the root of the destination oriented directed acyclic graph (DODAG). The DIOs propagate from the DODAG root through the network governed by the Trickle algorithm [Lev+11].

- Communication in the opposite direction is enabled by letting each node in the tree periodically send so-called DAOs to the DODAG root. That way, reverse routes are installed either at each intermediate node (storing mode) or exclusively at the root, which uses source routing to transmit to arbitrary nodes.

RPL has been extensively evaluated [TOV10; HC11; YCI13; CMN14; KG14; Ise+15] and is emerging as the protocol of choice for static low power and lossy networks.

2.2.5 LFFR

Observing the same high probability for datagram losses when using fragmentation that is discussed in Sect. 2.2.1, Thubert and Hui proposed a recovery mechanism for 6LoWPAN fragments at the 6LoWPAN layer itself within an IETF draft [TH14]. This protocol works with negative acknowledgments (NACKs), issued by the 6LoWPAN module at the IPv6 destination of a datagram. Every NACK includes a bit vector, which marks the missing fragments. NACKs are "routed" back to the origin of the datagram by using a reverse lookup and exchange of 6LoWPAN tag information that is stored at the nodes and carried in the NACK, respectively.

The enhancements to 6LoWPAN forwarding of fragments presented in this thesis could be combined with LLN fragment forwarding and recovery (LFFR) to further improve the reliability of transmissions of large datagrams. However, at the time of writing, the draft [TH14] has expired and has not been renewed so far.

2.2.6 6TiSCH

As described in Sect. 2.1, TSCH extension to IEEE 802.15.4 does not define any mechanism to actually create and deploy a communication schedule for a network. The IETF 6TiSCH working group was founded to tackle this problem and recently published a draft for the 6top protocol [WV16] and a general architecture of using 6LoWPAN over TSCH [Thu15].

2.3 Applications

6LoWPAN is mainly designed with the goal of transporting small datagrams using the IEEE 802.15.4 link layer. Its compression mechanisms reduce the overhead caused by IPv6 and upper layer protocol headers and for small payloads achieve to stuff an IPv6 datagram into a single IEEE 802.15.4 frame. However, given the minimum maximum MTU of 1280 B defined for IPv6, datagrams of this size are completely legal and therefore – I argue – will be used by future applications and should achieve an acceptable performance. Furthermore, there are already several applications from the field of wireless sensor networks which exhibit traffic patterns involving the transmission of large data objects.

A part of the advanced metering infrastructure, which in turn is a part of the efforts towards smart grids, is smart metering, i.e., the automated report of consumption data of customers to utilities. While in past days, this communication was restricted to the consumption values for a whole year, the realization of smart grids demands for detailed load profiles with a much finer temporal granularity. Such can come in sizes from several hundred bytes to several kibibytes, depending on the type of the meter. One option for the transport of such data are wireless mesh networks. Within the iEZMesh project [Wei+14a], requirements to transport load profiles of up to 3 KiB every 15 min were identified. While the proposed solution included a proprietary network protocol stack, a realization of such requirements with an IPv6 stack including 6LoWPAN is thinkable.

Another application scenario that potentially produces a large amount of data is structural health monitoring of buildings and structures like bridges, which has already triggered research efforts on transport protocols for large data objects [Pae+05;

Kim+07b; Kim+07a; PG10; Kur+12]. For these applications, data is created with high frequencies and sent to a sink for offline analysis. Approaches exists to reduce the data volume by shifting the computational effort to the sensor nodes and exchanging partial results among nodes [Boc+11], but have been tested only with very simple one-hop topologies. Similar with regard to the amount of data created are applications for earthquake, volcano monitoring [Wer+06; Suz+07] and image processing applications for, e.g., environmental monitoring [Ko+10].

In general, wherever "large" – in the sense of not fitting a IEEE 802.15.4 frame – data has to be transferred to a data sink, 6LoWPAN fragmentation has to be applied.

2.4 Energy Availability

For many applications in the field of wireless sensor networks or the IoT, a part of the participating nodes are powered by batteries or energy harvesters that do not yield enough for the node to be in an "always-on" state. A large number of research efforts for wireless sensor networks has therefore been directed at the development of low-power protocols, aiming at the extension of the lifetime of individual nodes or the whole network [MLT08; Bue+06]

Current available IEEE 802.15.4 transceivers consume a nearly equal amount of power in RX and TX state. For example, the Atmel AT86RF231 consumes 12.3 mA in RX and 14 mA in TX states according to the datasheet [14]. Put into sleep mode, on the other hand, most current transceivers draw current less than 1 µA. Therefore, to effectively reduce energy consumption, transceivers have to be duty cycled. Dominant techniques to achieve low duty cycles are low-power listening [Bue+06] and low-power probing [MLT08]. The former is usually implemented by either repeating the transmission of a frame multiple times (until the receiver wakes up and notices the activity) or by using long preambles. The latter takes an opposite approach: nodes ready to receive send a short probing frame to signal their readiness and senders stay active and listen for such a probe if they have something to send. Both approaches are mainly applicable for low-data rate applications – if nodes have to transmit continuously, the reduction in energy conservation is decreased. Also, low-power mechanisms trade savings in energy consumption with increased latency and decreased throughput due to waiting for other nodes' active phases.

To the best of my knowledge, there is no low-power listening or probing mechanism standardized for IEEE 802.15.4. According to [11a], nodes that act as routers in non-beacon enabled PANs have to be always on, because they can not know when a frame will be transmitted to them [Wat+16]. In beacon-enabled networks (Sect. 2.1), nodes can sleep in the contention free period during slots unused by them.

Various energy harvesting technologies have been proposed to allow for sustainable operation of sensor nodes [SK11]. Different technologies have been evaluated, ranging from photovoltaics over thermoelectric to piezoelectric. Harvesting solutions can be further differentiated by the type of storage system they use. Typically, at least one rechargeable battery is used. Additionally, supercapacitors, which allow for a near-infinite number of recharge cycles are integrated as primary storage to prevent many unnecessary shallow recharge cycles of the main battery [Ren13].

Most existing solutions aim at supporting nodes that are duty-cycled. However, with solar panels of a given dimension and a suitable environment, it is possible to

provide enough power for perpetual operation of always-on nodes. This has been demonstrated in a field test during the HelioMesh project, where solar panels were used to provide the energy for the motors of heliostats as well as an attached sensor node [Unt14].

Current IEEE 802.15.4 transceivers, e.g., the Atmel ATmega256RFR2, offer a low power RX state, cutting the current consumption by a factor of two. While the concrete working details of this mode of operation of the mentioned transceiver are not published by Atmel, I suspect that this mode implements a straightforward 50 % duty cycling scheme at the level of IEEE 802.15.4 symbols that activates the transceiver as soon as the preamble is received. It is claimed that activation of this mode decreases the receiver sensitivity by about 3 dB. While such transceiver supported energy saving modes can in their current state not be expected to replace more advanced low-power listening or probing techniques, they can help to reduce the overall energy consumption of routing nodes to allow for smaller dimensions of harvesters.

2.5 Goals of Evaluation

As stated in Sect. 2.2.1, the main goal of this dissertation is to evaluate the performance of 6LoWPAN fragmentation and the different forwarding strategies for large IPv6 datagrams. Furthermore, techniques to improve the situation are to be developed and their performance compared to the existing approaches. This is done for IEEE 802.15.4/6LoWPAN networks that use the non-beacon enabled mode of operation and assume that at least routing nodes within a network are always on. There are several reasons for this decision:

- The sending of large datagrams implies comparatively high data rates. For those, low-power listening or probing techniques are not well suited.

- At the time of writing, there is no standardized low-power mode of operation for IEEE 802.15.4, other than nodes sleeping in the CFP. The beacon-enabled modes, however, are not standardized for multi-hop operation until very recently with the proposal of DSME. Aiming at a fully standardized protocol stack, I decided to go for the unslotted operation with always-on nodes.

- Of the potential applications introduced in Sect. 2.3, smart metering can employ several grid-powered nodes (at electricity meters) and other applications (environmental, structural monitoring) are eligible for a backbone of routing nodes with generously dimensioned harvesting solutions.

Recent TDMA-based extensions to IEEE 802.15.4 (DSME, IPv6 over the TSCH mode of IEEE 802.15.4e (6tisch)) are promising alternatives to also provide good performance for large fragmented datagrams but are not considered in this dissertation. They yet have to prove that the proposed scheduling solutions work in large networks and with larger payloads [Wat+16].

A typical application scenario may employ the IETF stack described in Sect. 2.2.4. Fig. 2.4 shows a topology similar to those obtained from a testbed in the iEZMesh project. While during the project proprietary protocols were used, the network could also utilize an IETF stack and contain a number of always-on nodes that form a

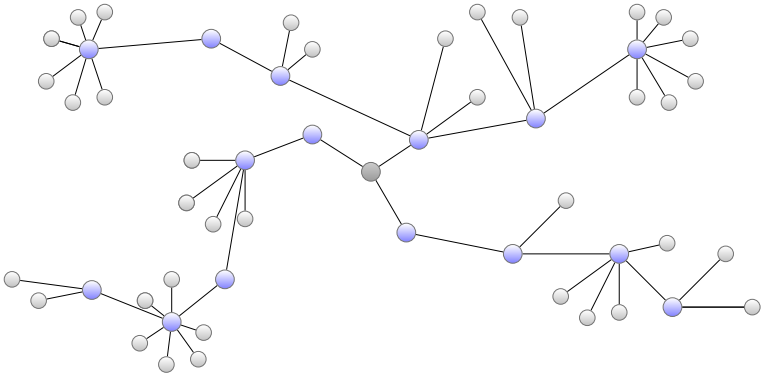


Figure 2.4: Application scenario with routing (blue) and leaf nodes (gray). The topology is similar to the routing topology observed during a field test in the iEZMesh smart metering project. The central node represents the wireless sink.

RPL routing tree, complemented by a number of RPL leaf nodes that do not actively participate in forwarding of data. Different from the routing nodes, these leaf nodes are expected to sleep most of the time and become active only when they have data to send. The extensions to IPv6 neighbor discovery for 6LoWPAN defines message formats and options to existing message formats to cope with such leaf nodes.

3 Analytic Model for 6LoWPAN-Fragmented Forwarding

This chapter introduces an extension to an existing analytical model for 6LoWPAN-fragmented transmissions of IPv6 datagrams.

3.1 Motivation and State of The Art

This section discusses the motivation for the creation of the model and provides an overview on existing related work.

3.1.1 Motivation

There are two major reasons to analytically model the process of 6LoWPAN fragment forwarding. First, for a fragmented datagram to be received successfully, all fragments belonging to that datagram have to be received successfully. It is intuitively clear that increasing the size of the datagram and hence the number of fragments decreases the probability for success on lossy wireless links, especially if transmissions via multiple hops are considered. Instead of relying on intuition, an analytical model can help to quantify this issue.

Secondly, between the two basic forwarding modes introduced in Sect. 2.2.3, there is a major difference in how they handle fragments of datagrams that get lost during reception. By definition, the Assembly mode will never forward any fragments of an incompletely received datagram: if any fragment gets lost, the datagram will not be sent to the IP layer and therefore will not be routed further. The Direct mode, on the other hand, forwards fragments as soon as they arrive. Thus, even if a datagram is lost early on the path, some of its fragments may propagate through the network without adding to the overall goodput. To quantify the magnitude of this effect is another motivation for creating a model.

Finally, there are two diametrically different approaches to handle the situation of a failed transmission of a fragment:

1. Assume the fragment and therefore the whole datagram is lost and in consequence, abort transmission of the whole datagram – this is called non-persisting strategy in the remainder of the thesis.

2. Continue sending fragments even after a transmission failure of a fragment and also continue sending subsequent fragments if a fragment is missing. In the former case, there is a probability that not the fragment, but the ACK was lost and therefore the datagram is not lost as a whole – this is called persisting strategy in the remainder of the thesis.

By means of an analytical model, these different approaches can be evaluated. I expect the persisting strategy to increase the success rate of datagrams marginally and at the same time to exhibit a comparatively large number of sent frames due to failed datagrams whose remains continue to propagate through the network. For this reason, the persisting strategy is not ideal for route-over schemes without any end-to-end recovery mechanism of fragments.

3.1.2 State of the Art

Several research efforts have been undertaken to analyze the performance of 6LoWPAN fragmentation over IEEE 802.15.4 networks using analytical models.

The model presented in Section 3.2 is derived from the model of Ayadi et al., who analyzed the efficiency of different TCP segment sizes ([AMR11]). Their model is based on bit error probabilitys (BEPs) that are stochastically independent. Fragmented datagrams, link-layer retransmissions, forward error correction and multiple hops are modeled. Typical effects of wireless channels like collisions, quality degradation or (self-) interference, on the other hand, are not captured by a channel model based solely on BEPs. The persistent strategy is used in case of failures. For a given set of input parameters, a datagram success rate and the expected number of bits sent can be determined with their model.

Subsequent work of Ayadi et al. [Aya+11] reuses their basic model and extend it to assess the effectiveness of LLN Fragment Forwarding and Recovery (LFFR), which is described within an internet draft of the IETF ROLL working group [TH14] and has expired at the time of writing. LFFR defines a layer-2.5 transport protocol for 6LoWPAN fragments, which enables the 6LoWPAN layer to retransmit individual fragments triggered by negative acknowledgments from the receiver. For this scenario, their assumption that nodes continue sending fragments although a former fragment has failed, is a reasonable one.

A different approach that tries to take into account collisions and is not specifically tailored to typical 6LoWPAN scenarios has been followed by Di Marco et al. [Di +12], which is in turn based on [Bia06]. Their models are based on Markov chains and model the transmission of frames over multiple hops by state transitions, taking into account clear-channel assessment, backoffs and link-layer retries. Output values of this model are latency, energy consumption (derived from a node's state) and reliability. This approach is potentially much more accurate than a model based solely on bit-errors, which neglects any potential and impacts of collisions between frames. However, their model is based on average rates of arrival and therefore can not capture the effect that a large number of frames arrive at a node simultaneously, as it is typically the case using fragmentation. The effect of a thus filled transmission queue is an increased probability for self-induced interference, which is neglected by the model (Sect. 2.2.3). This model was also extended by Meier and Turau, who added the handling of downstream traffic [MT15]. The extended model also recognizes the possibility for acknowledgments with data frames and with each other and take into account the increased probability of collisions after a collision (because two nodes start transmitting at the same time).

An analytical model that does take into account 6LoWPAN fragmentation and transmission of large datagrams was introduced by Ludovici, Di Marco, Calveras and Johansson [Lud+14]. It is used to compare the performance of CoAP's block-wise transfer [BS16], which is still being standardized at the time of writing, and

6LoWPAN fragmentation. Their results suggest that "6LoWPAN fragmentation outperforms CoAP blockwise transfer with regard to latency independently from the update generation rate and the number of nodes". CoAP blockwise transfer exhibits a higher reliability in congested traffic scenarios. The model is restricted to single-hop scenarios and therefore not applicable to the multi-hop scenarios considered in this thesis.

3.2 Model

The model is an extension of the one presented by Ayadi et al. and follows the same approach taken to derive probabilities and expectation values presented in their paper [AMR11]. It calculates the number of expected bits sent and the frame success probability (FSP) depending on link BEPs, the number of link-layer attempts and a forward error correction (FEC), i.e., a redundancy ratio which yields a number of correctable bit errors. It is assumed that bit errors are independent of each other. Additionally, it is assumed that duplicates are detected by the 6LoWPAN layer, which is realistic as it needs to keep track of the state of fragmented datagrams anyway.

As described in Sect. 3.1.2, the major difference between the existing model and the extension is the handling of failures and partial failures. While Ayadi et al. let a sender always send all fragments, I assume that a sender gives up on a datagram in case of a failure or partial failure. I also consider both forwarding techniques introduced in Sect. 2.2.3 in that context: Assembly and Direct mode. I implemented both the extended model and the original model of Ayadi. The latter was slightly modified to get comparable resulting quantities. Consistent with Sect. 3.1.1, the extended model for Assembly and Direct mode is referred to as non-persisting model, the original model is referred to as persisting model in the remainder of this thesis.

3.2.1 Link-Layer Model

At the link layer, three different outcomes of a transmission are defined:

- The transmission failed (probability $p_{\mathrm{f},k}$)
- The transmission partially failed, i.e., the frame arrived but the ACK was never received (probability $p_{\mathrm{p},k}$)
- The transmission was successful, frame and ACK arrived (probability $p_{\mathrm{s},k}$)

At their hearts, both models are based on the $p_{\mathrm{e},k}$ of a link k. In case of the occurrence of an uncorrectable bit error, the transmission is considered unsuccessful, yielding the introduced probabilities as

$$p_{\mathrm{f},k} = 1 - \sum_{i=0}^{c} \binom{L_{\mathrm{F}}}{i} p_{\mathrm{e},k}^{i} (1 - p_{\mathrm{e},k})^{L_{\mathrm{F}} - i} \tag{3.1}$$

$$p_{\mathrm{p},k} = (1 - p_{\mathrm{f},k})(1 - (1 - p_{\mathrm{e},k})^{L_{\mathrm{A}}}) \tag{3.2}$$

$$p_{\mathrm{s},k} = (1 - p_{\mathrm{f},k})(1 - p_{\mathrm{e},k})^{L_{\mathrm{A}}} = 1 - (p_{\mathrm{p},k} + p_{\mathrm{f},k}) \tag{3.3}$$

with the number of correctable bit errors c and the link layer frame sizes L_{F} and L_{A} for a data and Ack frame, respectively. I stick with the approach from [AMR11] to model FEC only for the data packet, not the acknowledgment. While the model itself thereby regards the possibility of a FEC mechanism, for the evaluation in this chapter I consider $c = 0$ and do not go into further details about the derivation of the number of fragments for a given datagram size.

Based on those basic formulas, the probabilities for success, partial failure and failure after r send attempts can be derived as

$$P_{\mathrm{s},k} = \sum_{j=1}^{r} p_{\mathrm{s},k}(1-p_{\mathrm{s},k})^{j-1} = \sum_{j=1}^{r} p_{\mathrm{s},k} \sum_{i=0}^{j-1} \binom{j-1}{i} p_{\mathrm{p},k}^{i} p_{\mathrm{f},k}^{j-1-i} \tag{3.4}$$

$$P_{\mathrm{p},k} = (p_{\mathrm{p},k} + p_{\mathrm{f},k})^{r} - p_{\mathrm{f},k}^{r} = \sum_{j=1}^{r} \binom{r}{j} p_{\mathrm{p},k}^{j} p_{\mathrm{f},k}^{r-j} \tag{3.5}$$

$$P_{\mathrm{f},k} = p_{\mathrm{f},k}^{r} \tag{3.6}$$

and the conditional expectation value for the number of sent bits in each case can then be derived by enumerating all possible outcomes after r attempts as

$$H_{\mathrm{s},k} = \frac{1}{P_{\mathrm{s},k}} \left(\sum_{j=1}^{r} p_{\mathrm{s},k} \sum_{i=0}^{j-1} \binom{j-1}{i} p_{\mathrm{p},k}^{i} p_{\mathrm{f},k}^{j-1-i} (jL_{\mathrm{F}} + (i+1)L_{\mathrm{A}}) \right) \tag{3.7}$$

$$H_{\mathrm{p},k} = \frac{1}{P_{\mathrm{p},k}} \left(\sum_{i=1}^{r} \binom{r}{i} p_{\mathrm{p},k}^{i} p_{\mathrm{f},k}^{r-i} (rL_{\mathrm{F}} + iL_{\mathrm{A}}) \right) \tag{3.8}$$

$$H_{\mathrm{f},k} = \frac{1}{P_{\mathrm{f},k}} rL_{\mathrm{F}} P_{\mathrm{f},k} = rL_{\mathrm{F}} \tag{3.9}$$

$$H_{\mathrm{sp},k} = \frac{1}{1-P_{\mathrm{f},k}} \left(P_{\mathrm{p},k} H_{\mathrm{p},k} + P_{\mathrm{s},k} H_{\mathrm{s},k} \right) \tag{3.10}$$

where $H_{y,k}$ is the expected number of bits sent in case of **s**uccess, **p**artial failure or **f**ailure on link k, with the corresponding probability $P_{y,k}$. In addition to the values for the three possible outcomes, I define as $H_{\mathrm{sp},k}$ the expected number of bits sent in case of a success or a partial failure. The formulas presented so far are basically those introduced by Ayadi et al. [AMR11], extended by an index to identify a certain hop and a slight rearrangement due to the different results which partial failures produce in the multi-hop model.

3.2.2 Multi-Hop Model

We define the route the fragments have to traverse as a series of hops (wireless links) k numbered from 1 to n. Moving to this multiple hop, multiple fragment scenario, we get for the probability of success and failure of a datagram consisting of m fragments

$\mathcal{P}_\mathrm{s}$ and $\mathcal{P}_\mathrm{f}$, independent of the actual forwarding mode:

$$\mathcal{P}_\mathrm{s}(h_0,h,m) = \prod_{k=h_0}^{h} (P_{\mathrm{s},k}^{m-1}(P_{\mathrm{s},k}+P_{\mathrm{p},k})), \tag{3.11}$$

$$\mathcal{P}_\mathrm{f}(h_0,h,m) = \sum_{k=h_0}^{h} \mathcal{P}_\mathrm{s}(h_0,k-1,m)\left(\sum_{x=1}^{m-1} P_{\mathrm{s},k}^{x-1}P_{\mathrm{p},k} + \sum_{x=1}^{m} P_{\mathrm{s},k}^{x-1}P_{\mathrm{f},k}\right), \tag{3.12}$$

where h_0 is the index of the first hop, h the index the final hop and m the number of fragments sent. For example, $\mathcal{P}_\mathrm{s}(1,8,10)$ is the probability of success for a datagram fragmented into 10 fragments on a route of 8 hops, the bit error probabilities $p_{\mathrm{e},k}$ have to be defined accordingly for each link. For the last fragment sent, a partial failure is sufficient, because it does not matter whether the sender would give up on the datagram afterwards.

With the Assembly mode, no fragments of a failed datagram are propagated any further after the first fragment failure. The expected number of bits sent in Assembly mode E^A therefore is

$$E^\mathrm{A}(h_0,h,m) = \mathcal{P}_\mathrm{s}(h_0,h,m)E_\mathrm{s}^\mathrm{A}(h_0,h,m) + \mathcal{P}_\mathrm{f}(h_0,h,m)E_\mathrm{f}^\mathrm{A}(h_0,h,m) \tag{3.13}$$

$$E_\mathrm{s}^\mathrm{A}(h_0,h,m) = \sum_{i=h_0}^{h} ((m-1)H_{\mathrm{s},i} + H_{\mathrm{sp},i}) \tag{3.14}$$

$$\begin{aligned} E_\mathrm{f}^\mathrm{A}(h_0,h,m) = {} & \frac{1}{\mathcal{P}_\mathrm{f}(h_0,h,m)} \sum_{k=h_0}^{h} \mathcal{P}_\mathrm{s}(h_0,k-1,m)\times \\ & \left(\sum_{x=1}^{m-1}((x-1)H_{\mathrm{s},k} + H_{\mathrm{p},k} + E_\mathrm{s}^\mathrm{A}(h_0,k-1,m))P_{\mathrm{s},k}^{x-1}P_{\mathrm{p},k}\right. \\ & \left. + \sum_{x=1}^{m}((x-1)H_{\mathrm{s},k} + H_{\mathrm{f},k} + E_\mathrm{s}^\mathrm{A}(h_0,k-1,m))P_{\mathrm{s},k}^{x-1}P_{\mathrm{f},k}\right) \end{aligned} \tag{3.15}$$

where E_s^A and E_f^A are the conditional expectation values of the number of bits sent in case of success and failure, respectively.

While the conditional expectation value of the number of bits sent in case of success is the same for the Direct mode, for the case of a failure all fragments that have already been transported to the next hop along the route have to be taken into account and their contribution to the overall number of bits sent has to be considered. To include all those possible outcomes, I define $E^\mathrm{D}(h_0,h,m,H_\mathrm{acc},P)$ by means of a recursive formula. In order to simplify presentation and to foster understandability, I omit the parameters h_0, h, m, k, and H_acc in formulas for E^D, M_p, M_f, and H_z in equations (3.16) to (3.19). M_p and M_f contain the actual recursion:

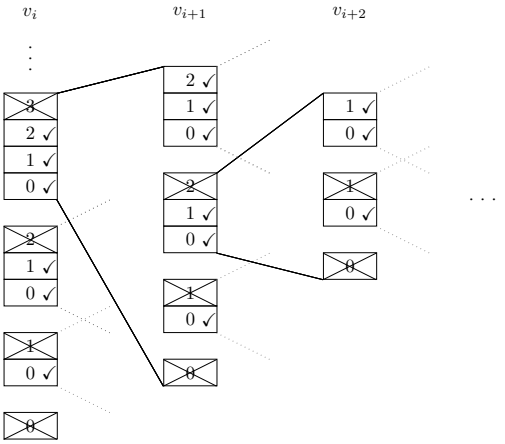


Figure 3.1: Illustration of cases to consider for Direct forwarding in case of a transmission failure of a fragment. At node v_i, transmission failures of fragments 0 to 3 is shown. A Failure at fragment 3 means, that at node v_{i+1} a partial datagram of 3 fragments will be transmitted, which can again fail at fragments 0 to 2 or succeed, leading to a transmission of two fragments at node v_{i+2} and so on. Dotted lines indicate the missing cases not explicitly shown.

$$E^{\mathrm{D}} = \begin{cases} PH_{\mathrm{acc}}, & h < h_0 \vee m = 0 \\ P\mathcal{P}_{\mathrm{s}}(h_0, h, m)\left(E_{\mathrm{s}}^{\mathrm{A}}(h_0, h, m) + H_{\mathrm{acc}}\right) + P\sum_{k=h_0}^{h}\left(M_{\mathrm{p}} + M_{\mathrm{f}}\right), & \text{else} \end{cases} \tag{3.16}$$

$$M_{\mathrm{p}} = \sum_{x=1}^{m-1} E^{\mathrm{D}}(k+1, h, x, H_{\mathrm{p}}, \mathcal{P}_{\mathrm{s}}(h_0, k-1, m) P_{\mathrm{s},k}^{x-1} P_{\mathrm{p},k}) \tag{3.17}$$

$$M_{\mathrm{f}} = \sum_{x=1}^{m} E^{\mathrm{D}}(k+1, h, x-1, H_{\mathrm{f}}, \mathcal{P}_{\mathrm{s}}(h_0, k-1, m) P_{\mathrm{s},k}^{x-1} P_{\mathrm{f},k}) \tag{3.18}$$

with

$$H_z = H_{\mathrm{acc}} + (x-1)H_{\mathrm{s},k} + H_{z,\mathrm{k}} + E_{\mathrm{s}}^{\mathrm{A}}(h_0, k-1, m), z \in \{\mathrm{p}, \mathrm{f}\} \tag{3.19}$$

As for the Assembly mode, the formula sums up the expectation values for the link layer transmission multiplied by the corresponding probability for failures at a certain hop and a certain fragment. However, a failure of the xth fragment on the kth hop means a partial datagram of $x-1$ (or x in case of partial failure) fragments will be send from hop $k+1$ to hop h (Fig. 3.1). This is captured by applying the recursion, additionally passing the accumulated expectation value of the number of bits sent so far (parameter H_{acc}) and passing the product of corresponding probabilities (parameter P). Thereby, all possible cases are enumerated. In the initial formula H_{acc} and

Table 3.1: Default parameter values

Parameter	Value	Parameter	Value
r	5	c	0
h_0	1	h	8
m	12	$p_{\mathrm{e},k}$	2.4244×10^{-4}

P are set to 0 and 1, respectively. For example, the expected number of bits sent to send 10 fragments via hops 1 to 5 then can be obtained by $E^{\mathrm{D}}(1, 5, 10, 0, 1)$.

3.3 Evaluation

To evaluate the impact of the different forwarding techniques, the model is fed with different scenarios varying BEP, denoted as p_e , number of link-layer attempts, number of hops and number of fragments. The potential use of FEC ($c > 0$) is not evaluated further. For all scenarios, $L_{\mathrm{F}} = 119\,\mathrm{B}$ is used, including 11 B MAC header and 2 B PHY header, leaving an 802.15.4 payload of 106 B. A link-layer acknowledgment of 7 B size is assumed. Unless specified differently, I set the non-varying input parameters to the values shown in Table 3.1. Besides the overall probability of success for a transmission, the model's main output metric is the expected number of bits sent, which indicates the amount of energy used for transmission as well as the overall traffic load produced. In the following, additional subscripts P and NP indicate persisting and non-persisting forwarding strategy (see Section 3.2), respectively.

While the model allows for a different BEP to be assigned to each link, for the evaluation presented here I used equal rates for all links. To calculate results, the model was implemented as a Mathematica module. Results were calculated and compared with different guarantees for precision, so that numerical instabilities biasing the results can be ruled out.

3.3.1 Persistent vs. Non-Persistent

First, the difference between persisting and non-persisting forwarding is assessed by comparing the output from the original model with the extended version. On the one hand, I expected the probability of success for the persisting strategy to be slightly higher than for the Assembly and non-persisting Direct mode, because partial failures on a link are counted as an overall success with regard to the transmission of a fragment. On the other hand, the expected number of bits sent should be significantly higher for the persisting approach, as even in the case of a failure a sender continues sending all remaining fragments. Figure 3.2 shows a comparison of the two strategies for different values of the BEP. Subscripts P and NP denote quantities for persisting and non-persisting strategy, respectively.

While the effect of persisting on the probability of success is comparatively small – for the shown BER values it stays below 0.03 –, the impact on the expected number of bits sent is significant. Note also that in a real network, failures are likely caused by interfering transmissions of other nodes. Continuing transmissions after a failure may therefore also impact other, not-yet-failed, transmissions. In the light of those

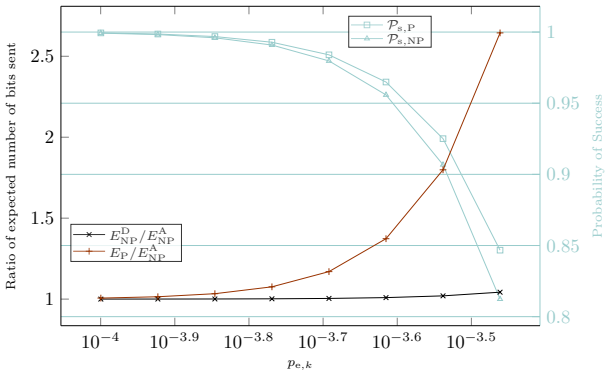
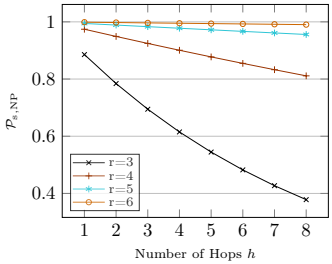


Figure 3.2: Comparison of the ratios of the number of expected bits sent (left y-axis) and the probability of success (right y-axis) for persisting (subscript P) and non-persisting (subscript NP) approaches against the BEP (p_e).

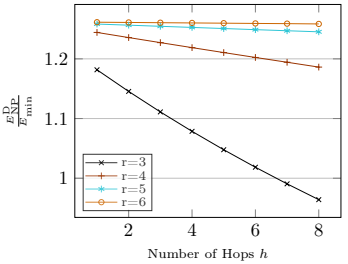
results, using such a persisting strategy for the given scenario is considered inadvisable. Therefore, in the remaining evaluation, the focus is shifted to the non-persisting method.

3.3.2 Multi-Hop Transmissions

Figure 3.3a indicates the impact of the number of hops and retransmissions for applying 6LoWPAN fragmentation: It shows the probability of successful transmission



(a) Probability of success $\mathcal{P}_{s,NP}$ against number of hops for different maximum numbers of retransmissions



(b) Normalized expectation value of number of bits sent E^{D}_{NP}/E_{min} against number of hops for different maximum numbers of retransmissions

Figure 3.3: Influence of number of hops and the maximum number of retransmissions, non-persistent mode

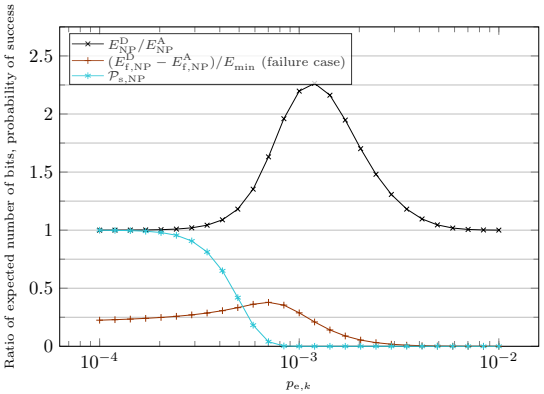


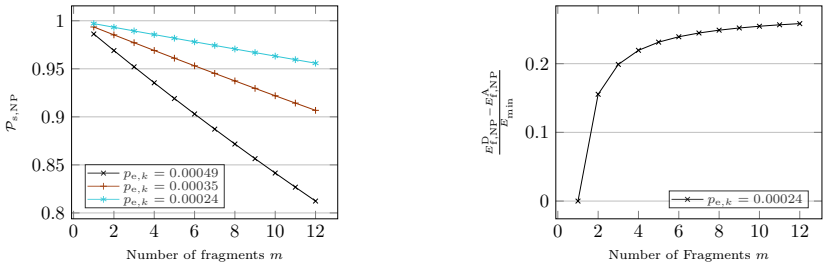
Figure 3.4: Non-persisting mode; Ratio of expected number of bits sent in Direct ($E^{\mathrm{D}}_{\mathrm{NP}}$) and Assembly mode ($E^{\mathrm{D}}_{\mathrm{NP}}$), together with overall probability of success and the difference in bits for the number of bits sent in case of failure for both modes, normalized by E_{min}, against BEP.

$\mathcal{P}_{\mathrm{s,NP}}$ of a large, 6LoWPAN-fragmented datagram for one to eight hops and for different maximum numbers of link-layer retransmissions r. It can be seen that $\mathcal{P}_{\mathrm{s,NP}}$ decreases for an increasing number of hops. The effect is more strongly pronounced for smaller values of r. For the given p_e and $r = 3$ (the default value defined in the IEEE 802.15.4 standard), $\mathcal{P}_{\mathrm{s,NP}}$ drops below 0.4 at eight hops. A higher max. number of link-layer retransmissions counters this effect: for $r = 6$, $\mathcal{P}_{\mathrm{s,NP}}$ only drops a few percent below 1.

Additionally considering the expected number of bits sent (normalized by the minimum number of bits needed for the complete transmission), shown in Fig. 3.3b, it can be seen that the improved probability for success is bought by increasing the number of frames sent. For $r = 6$, more than 25 % more bits than for a "perfect" run are sent. The increased number of bits does not have any effect on the probability of success in the presented model. In a real network, this increased traffic volume may increase the probability of collisions and therefore potentially reduce the positive effect on the overall probability of success of increasing r. Still, the result indicate that a larger number of retransmissions in general increases the probability of success.

3.3.3 Additional Bits in Direct Mode

Figure 3.4 shows the ratio of expectation values of bits sent for Direct and Assembly modes ($E^{\mathrm{D}}_{\mathrm{NP}}/E^{\mathrm{A}}_{\mathrm{NP}}$) along with the overall probability for success of the whole datagram $\mathcal{P}_{\mathrm{s,NP}}$, which is the same for both forwarding techniques. For success rates approaching 1 and 0, the difference between the expected number of bits sent in Direct and Assembly forwarding modes approaches zero. However, for a probability of success of 0.81 a ratio of 1.04 can be observed, i.e., a 4 % increase compared to the



(a) Probability of success

(b) Normalized difference between Direct and Assembly mode

Figure 3.5: Influence of number of fragments (datagram size) for several values of $p_{e,k}$ and $r = 5$

Assembly mode. With lower probabilities of success, an even higher ratio E^{D}_{NP}/E^{A}_{NP} can be observed.

This difference is exclusively caused by the direct mode's conditional expectation value of bits sent in case of a failure. To better illustrate this increased number of bits in direct mode in case of a failure, I added the difference of the corresponding conditional expectation values of Direct and Assembly mode to the plot, normalized by the minimum number of bits needed for a successful transmission ($(E^{D}_{f,NP} - E^{A}_{f,NP})/E_{min}$). In case of failure, the direct mode produces about one fourth of the minimum number of bits needed for successful transmission more than the Assembly mode. The probabilities for success being equal, this difference is caused exclusively by fragments of datagrams that have already been lost at some node.

The normalized difference between Direct and Assembly mode in case of failure ($(E^{D}_{f,NP} - E^{A}_{f,NP})/E_{min}$) is also evaluated against the number of fragments m (Fig. 3.5b). As expected, there is no difference for only a single fragment – Direct and Assembly behave exactly in the same way for this case. With the number of fragments increasing, the normalized difference increases, though with a decreasing slope.

It can also be seen that for an increase in the number of fragments, the probability of success decreases (Fig. 3.5a). The higher the BEP, the more pronounced is the effect. For higher BEP, even the used parameter setting of $r = 5$ yields probabilities of success for a large number of fragments approaching 0.8.

3.4 Conclusions

This chapter presents an analytical model based on the BEP of a link. Due to the fact that it does neither model channel contention nor (self-) interference nor issues like queue sizes and buffer drops, its accuracy is clearly limited. However, for a given bit-error rate, the quantitative results can help to estimate the impact of important parameters on 6LoWPAN-fragmentation.

Given a significant probability for errors, which is a usual case for real wireless networks, it shows that the combination of a large number of fragments and and a large number of hops dramatically decreases the overall probability of success of a datagram transmission and is thereby consistent with the my expectation. Increasing the number of retries counters the effect, albeit at the cost of additional frames sent. If those additional frames can actually have a negative impact on the overall datagram success rate is evaluated by simulation and testbed experiments (Chapter 5 and Chapter 7).

It could also be seen, that a persisting strategy on the one hand increases the probability of success slightly, but on the other hand produces a large number of additional fragments in the network in comparison to the non-persisting strategy. Furthermore, it was shown that in case of failure, the Direct mode creates about 25 % more fragments than the Assembly mode. These additional fragments do not contribute to the goodput of the network. However, the share of these fragments in the total number of fragments is small for datagram success rates of more than 80 % and therefore is not expected to have a significant influence on the performance in real networks.

4 Simulation Model and Environment

The proposed enhancements to a 6LoWPAN implementation (Sect. 5.2.1), including 6LoOF (Sect. 8), are evaluated by simulations and testbed experiments. Simulations were mainly used to perform parameter studies that were too expensive in terms of time for testbed experiments. To obtain meaningful results, the model used by the simulation environment has to reflect properties of the real world that have an impact on the behavior of the simulation. This means that a reasonably accurate model for the IEEE 802.15.4 physical layer operating in the 2.4 GHz ISM band has to be implemented. Furthermore, the simulation model has to be validated [WGG10]. This chapter introduces the used simulation tools and modeling frameworks (Sect. 4.1). It furthermore discusses methods to model the physical layer of wireless transmissions and describes the one taken for this thesis (Sect. 4.2). Lastly the approach taken to produce simulation scenarios from measured data to validate the simulation model (Sect. 4.3) is described.

4.1 Frameworks and Tools

This section introduces the simulation environment and frameworks that were utilized to obtain the measurement data.

4.1.1 OMNeT++

The OMNeT++ simulation framework [Var99; Var+01; VH08] has been around since 1997. While being a generic "discrete event simulation environment", it is mainly used for simulation of wired, wireless and mobile communication networks. OMNeT++ itself does not contain any simulation models. Support for a large number of internet protocols and more recently also for an increasing number of wireless protocols and models for the physical layer is provided by the INET model suite.

OMNeT++ is based on modules, which communicate with each other by passing messages to gates. The simulation model is then defined by connecting the gates between modules and the C++-implementation of the individual modules. By adhering to this general concept, a loose coupling of reusable components can be realized.

Various tools are available for debugging, verification and analysis. These reach from "watching" network packets in slow motion within the provided graphical user interface over detailed event logs capturing all message passing events to statistics recording and graphical analysis of those.

I chose OMNeT++ as simulation environment, because of its openness for academic purposes, the described modular design and availability of several extensions for advanced modeling of the physical layer. Furthermore, another open and popular simulation environment, ns3 [RH10], was not as well developed at the time the decision was made.

4.1.2 MiXiM

One thing lacking from the OMNeT++/INET-combination until recently, were sophisticated models for the physical layer. Among the available solutions providing those was the MiXiM framework [Köp+08; Wes+09], which also contains models for mobility, batteries and mac protocols. Castalia [PTB10] is another simulator, which is based on OMNeT++'s event engine and which is focused on PHY layer models for wireless sensor and body area networks. For this thesis and CometOS (Sect. 4.1.3), MiXiM was chosen, because it easily integrates into existing OMNeT++ projects and provides sophisticated means to provide physical models as well as a basic IEEE 802.15.4 link layer implementation. At the time of writing, both the original MiXiM project and Castalia seem to be abandoned, but parts of MiXiM are being refactored and integrated into INET.

MiXiM's physical layer simulation is realized by a customizable (by inheritance) `BasePhyLayer` module. This module is responsible for the transmission and reception of `AirFrames`. When transmitting an `AirFrame`, it is passed to all potential receivers and nodes within interference range of the sender. The actual receivers are determined by a `ConnectionManager` to avoid expensive calculations for nodes that are not within interference range. Reception then comprises three steps:

- Pass the `AirFrame` to a `ChannelInfo` object, which keeps track of all incoming frames for a node.
- Pass the `Signal` to all active `analogueModels` for processing
- Pass the `AirFrame` to a `Decider`.

The latter two actions offer entry points for customization. First, the type of the `Decider` can be configured and it is also possible to create new `Deciders` by creating a new subclass. The `Signal`, which is attached to the `AirFrame`, is processed by `analogueModels`. A `Signal` consists of information about the signal strength (depending on the node's transmission power) and information about its temporal extent and its spatial origin. Each `analogueModel` calculates an attenuation and attaches it to the `Signal`, resulting in a `Mapping` (a matrix-like data structure) that represents signal strength at the receiving node for different points in time and optionally different frequencies.

The actual decision about whether a frame is received correctly then is delegated to the decider by calling it at least at the beginning and end of a frame transmission. The more sophisticated `Deciders` bundled with MiXiM check if the strength of an incoming signal is strong enough to be received at all and if another signal is already being received (in which case, the new signal is not received). "Strong enough" here means that during the SFD, the SINR of the incoming signal does not cause a bit error. To calculate the SINR, all incoming and processed (attenuated by the `analogueModels`) `Signals` stored in the `ChannelInfo` object are considered. At the end of a frame reception, the decider is called again and determines if a bit error has occurred within the frame. The calculation of bit error probability depends on the used modulation technique and the SINR. The MiXiM-provided `Deciders` then either pass the received frame or the information that the frame was dropped to the connected MAC module.

Thereby, MiXiM provides an simulation of the physical layer at the bit level [WGG10]. It is also able to model collisions of frames by means of the `ChannelInfo` module.

4.1.3 CometOS

The protocols evaluated in this thesis are implemented in CometOS. CometOS is – similar to OMNeT++ – based on modules that exchange messages via gates. This similarity is deliberate. Several adapter classes enable modules written for CometOS to be executable as OMNeT++ simulations. Thereby, CometOS aims to provide a framework for rapid protocol development and verification.

In addition to the simulation capabilities via OMNeT++, CometOS provides a core framework that provides modules, gates and message passing, as well as a non-preemptive scheduler for arbitrary platforms. A platform-dependent implementation of a very slim platform abstraction layer (PAL) is required for the core functions to work. Additional PALs define interfaces for typical communication buses (UART, TWI, SPI), low-level IO (GPIOs), watchdog timer, persistent memory and a link layer for wireless communication. The latter is used by a platform-independent module named MacAbstractionLayer, which offers a unified interface for protocol stacks running within an OMNeT++ simulation or a specific hardware platform.

We chose the link layer as level of abstraction between simulation and hardware-based platforms, because several transceivers already support extended operation modes, which implement the complete CSMA/CA protocol and automatic acknowledgments and retransmissions in hardware and therefore provide a good match for this level of abstraction. Relying on such extended operating modes proved to be hazardous, as explored in Chapter 6.

Additional features of CometOS are support for cross-layering by attaching meta information to messages and support for simple synchronous and asynchronous remote function calls. Furthermore, interface files and typemaps are provided to facilitate the creation of a Python interface and language bindings using the software interface generator (SWIG) [Bea96]). Thereby, an existing protocol stack implementation can be augmented by Python scripts to create a powerful "basestation" that can be employed to control a wireless network or run automated experiments within a network. This approach has been used heavily during the evaluation phase of this thesis (Sections 5.4 and 8.4).

CometOS was designed with the domain of wireless sensor networks in mind and therefore is lightweight enough to run on resource-constrained embedded hardware (e.g., Atmel ATmega128RFA1). Contrary to other popular lightweight operating systems for wireless sensor networks (TinyOS [Lev+05], Contiki [DGV04]), CometOS also allows and makes use of dynamic memory allocation, similar to RIOT OS ([Bac+13]).

4.2 Physical Layer Model

There are numerous models available to simulate the physical layer of wireless networks, which can be distinguished by the aspect of wireless channels they model. This section discusses several approaches and describes the approach taken for the simulation studies carried out in this thesis.

4.2.1 Available Models for Wireless Sensor Networks

Empirical path loss models determine the signal strength observed by a receiver dependent on the distance between sender and receiver, antenna gains and a set of

parameters that have to be set according to environmental factors – line of sight or not, indoor, outdoor, urban or rural area, etc., are typical environmental factors [Sey05; Rap02]. Pure path loss models do not have a stochastic component, i.e., will always produce the same results for the same set of parameters.

Log-normal shadowing models add a stochastic component to the attenuation of path loss models. It models the obstruction of transmission paths by large objects, resulting in varying signal strengths for equal distances, known as flat fading. They have also been shown to reflect the behavior for indoor wireless sensor networks with good accuracy [NH93; ZK04; Rap02]. An example is the log-distance path loss model [Rap02], which combines a deterministic path-loss with a stochastic log-normal shadowing component.

Stochastic fading models try to capture the effects caused by multi-path propagation of wireless signals. A signal traveling along different paths due to reflections may cause self-interference at the receiver. Additionally, moving transmitters and/or receivers are subject to the Doppler spread. These effects can lead to attenuation of the signal (fading). Rayleigh and Rician distributions can be used to model different categories of fading [Sey05].

MiXiM defines `analogueModels` of each category: `SimplePathloss`, `LogNormalShadowing` and `JakesFading` (which is an implementation of Rayleigh fading). A fundamentally different approach is based on deriving received signal strengths from recorded measurements within real testbeds.

Lee et al. [LCL07] examine this possibility to use actual real-world traces of samples of the received signal strength indicator (RSSI) to more accurately model noise. The authors measure signal strengths of noise with a CC2420 radio transceiver to create noise traces. Different methods to produce an actual noise value within a simulation based on those traces are introduced and evaluated, ranging from naive sampling over an algorithm they called closest-fit pattern matching to correlation distortion [Joh94]. In conclusion, the closest-fit pattern matching method is identified as a promising candidate to especially model time-dependent interference more accurately.

The idea to base the physical layer on traces of real-world networks is carried farther by Rusak and Levis [RL08], who extend the general idea to traces of the signal strength of received data frames rather than sampling noise. Basically, the same closest-fit pattern matching (CPM) algorithm is applied. It is augmented by algorithms to estimate the signal strength for frames that were not successfully received and by algorithms to correct for the noise that is measured in addition to the signal strength at reception of a frame. Their approach acknowledges the fact that, in real networks, a correlation exists between the number of consecutive successful or failed frame receptions and the probability of success for the next frame. The “classic” stochastic models on the other hand, consider consecutive frame transmissions as independent.

As metric to measure the accuracy of either model, the Kantorovich-Wasserstein distance [GS84] between the conditional packet delivery [Sri+10] functions is employed. The authors claim that their approach significantly increases accuracy of the simulation of a physical layer. Especially for a low-PRR link, the chosen metric shows a much better match of their CPM approach in comparison to a log-normal shadowing and constant signal strength simulation techniques. However, results are only shown for so-called intermediate links, which exhibit packet reception rates of 58.5 % and 82.5 %. This distorts the result, because, good links, which account for a large part

of the links, depending on which study of wireless transmissions is consulted, exhibit a much less time-variant behavior [Sri+10], and hence can be modeled accurately by the log-normal shadowing and constant signal strength approaches.

4.2.2 Choosing an Appropriate Model

The subject of examination for this thesis has been stated as the performance of different forwarding strategies for 6LoWPAN-fragmented datagrams. The simulation model has to reflect important aspects of wireless communication accurately. At the same time, characteristics that are not important, should be abstracted from [WGG10].

Path Loss

This thesis is not concerned with the prediction of the performance of a 6LoWPAN-enabled wireless network based on the real world position of its nodes, i.e., to guide and prepare its deployment or to determine if it can fulfill certain application requirements. Therefore, I argue that empirical path loss models that convert positions of nodes and a large parameter set into a distribution of received signal strengths do not add any specific value to the subject of examination. In contrast, they even complicate the creation of a meaningful network topology and its validation. Most path loss models require configuration by various parameters (Sect. 4.2.1). Thereby, validation of the simulation model is complicated by the task to first tune those parameters to the environment of a potential testbed.

Collisions and Channel Contention

Considering multi-hop communication paths, hidden-terminal collisions and channel contention are expected to have tremendous impact on the overall performance. Fragmentation of datagrams implies that a sender will transmit multiple IEEE 802.15.4 frames in direct succession and thereby give rise to self-interference along the path. Modeling collisions reasonably well is therefore considered of high importance.

Variation of Received Signal Strength

Even in static environments, i.e., networks of position-fixed nodes, the link quality may vary between frames. Moving people and objects can slightly alter propagation paths or shadow a dominant path and thereby change the wireless channel [Bac+12]. Analogous to the argument against modeling path loss, one may argue, that even for a constant distance/received signal strength between fixed nodes, a random factor can be introduced by calculating bit error rates from the signal-to-noise ratio. Hence, randomization of the RSSI would not add any new aspect to the model. However, this reasoning neglects that without some stochastic component, the SINR for interfering transmissions of any pair of two nodes in the network will then also be constant.

4.2.3 A Measurement-Based Physical Layer

In consequence of the discussion in Sect. 4.2.2, I chose to use a PHY layer simulation model that essentially mimics log-normal shadowing, but instead of using some path

loss model to calculate average received signal strength, the average RSSI value can be configured directly for each link. This allows the easy creation of simple "ideal" networks, as well as the creation of simulation scenarios derived from measurement data collected in real testbeds. Furthermore, comparison of evaluation results collected by simulation and experiments in the same testbed can serve as a validation for the overall simulation model.

The signal strength stays constant during the duration of a single frame and is only randomized once at the start of the reception using the provided log-normal distribution. An SINR is calculated between the signal under investigation and all other signals on the same IEEE 802.15.4 channel. Adjacent or alternate channel interference is not considered, because the adjacent channel rejection of current transceivers is good enough ($> 32\,\mathrm{dB}$) for the issue to be of no practical importance as long as interferers are not situated in very close proximity to a receiver [Wei10].

The chosen approach does not take into account temporal correlation of packet delivery success and failure like the one introduced in Sect. 4.2.1. As discussed, such temporal correlation is mainly a concern for the class of intermediate links and hence is an important factor to consider especially for link quality estimation and routing decisions. However, the scope of the studies presented in this thesis is 6LoWPAN fragmentation, which takes place between the link layer and the routing layer.

Carrying this argument farther, it was also decided to use a static routing scheme at the network-layer. On the one hand, this means that if routes are lost due to temporal degradation of a link, the performance during an experiment run will suffer greatly. On the other hand, changing of routes in a network also has non-negligible influence on the overall performance, albeit to a less extent than a disconnected network. There even exist proposals to define some topology metric that reflects the capabilities of the network for any given time [Puc+10] to counter such issues when comparing the performance of different routing protocols in real testbeds.

Both approaches – static and dynamic routing – cannot prevent changes in the underlying real network topology. Using a static routing scheme, a disconnected network can be detected relatively easy and can then be removed from the set of runs. Being not interested in the routing protocol itself and recognizing that a parametrizable protocol like RPL [Win+12] does in itself add to the overall parameter space, I went with a static routing scheme at the network layer.

Another reason for choosing the simpler log-normal stochastic model is a more practical one. The widespread (e.g., M3 OpenNode of the IoTLab) Atmel AT86RF231 transceiver only reports RSSI and energy detection (ED) values of down to $-90\,\mathrm{dBm}$, which is 10 dB above its sensitivity. Reported RSSI values of $-90\,\mathrm{dBm}$ for incoming packets therefore do not carry very much information. The method to collect trace-based data described in Sect. 4.2.1 is therefore difficult to apply with that transceiver.

Further properties of the simulation model are described by means of its implementation for the MiXiM framework, version 2.1. To facilitate an approach with user-defined link properties, existing classes were extended by inheritance (Fig. 4.1). A simplified UML sequence diagram shows the collaboration between those classes for the reception of an `AirFrame`, which is similar to the description given in Sect. 4.1.2. Main difference to the default MiXiM behavior is the introduction of the `EmpiricDeciderBase` class. This class adds an attenuation mapping corresponding to the configured link

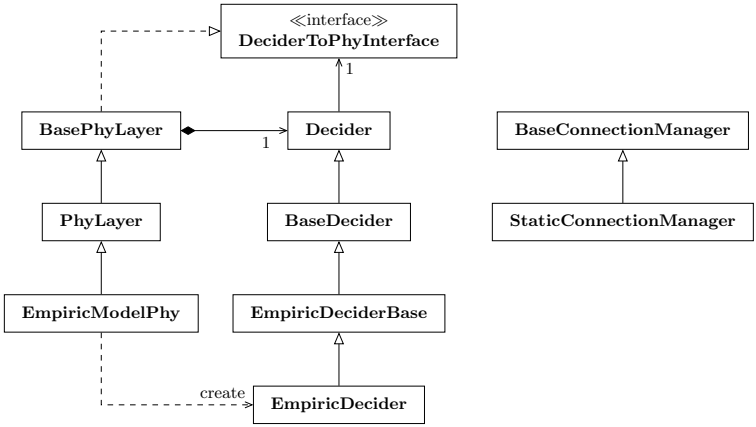


Figure 4.1: Extension of MiXiM classes

quality to the signal of an `AirFrame` the first time it is processed. Delegating the responsibility to add the attenuation mapping to an `analogueModel` would be preferable, but has not been possible with MiXiM version 2.1, as those are only passed the signal itself, but not the `AirFrame`, from which source and destination address can be derived.

The actual decision about receiving or dropping of the `AirFrame` is then performed in `processSignal` methods. The model operates at the bit domain and derives a bit error probability for every bit from the S(I)NR of the signal according to

$$BEP = \frac{8}{15} \times \frac{1}{16} \times \sum_{k=2}^{16} -1^k \binom{16}{k} e^{\left(20 \times SINR \times \left(\frac{1}{k} - 1\right)\right)}, \tag{4.1}$$

as defined in [11a]. The given formula does take into account the direct-sequence spread-spectrum technique and the O-QPSK modulation IEEE 802.15.4 defines for the 2.4 GHz ISM bands. It further assumes that interference can be modeled as additive white Gaussian noise, which is reasonable for interferers like those based on IEEE 802.11, which use a larger bandwidth than IEEE 802.15.4. As discussed by Son et al. [SKH06], the assumption is not entirely accurate for interference by other IEEE 802.15.4 transmissions.

Available IEEE 802.15.4 transceivers operating in the 2.4 GHz band are not able to transmit and receive frames at the same time. MiXiM also reflects this behavior by adding an attenuation to the signal that results in an S(I)NR of 0. Using (4.1) to calculate the bit error rate, this S(I)NR yields a bit error probability of p_{BER}=0.5. This is reasonable, given that additive white Gaussian noise (AWGN) may randomly produce a correct value even in the absence of any signal. However, this results in a non-negligible probability that an SFD is received with the radio being in TX state, causing the MAC layer module to enter inconsistent states (or to assert). Therefore, an additional check for an S(I)NR of 0 was added to the decider to prevent the simulation model from ever receiving a frame while being in transmission state.

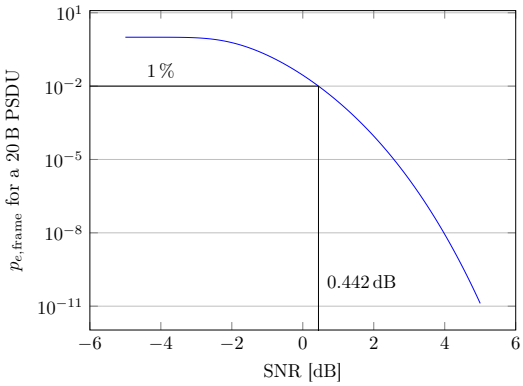


Figure 4.2: Determining the thermal noise for a transceiver sensitivity of −100 dBm

As no further forward error correction or bit interleaving is used in IEEE 802.15.4, the bit error probability is directly applied to the part of an incoming frame under examination. The `EmpiricDecider` does not perform an explicit CRC calculation, but rather assumes that a bit error leads to a CRC failure and hence a dropped frame. Thus, if a bit error occurs, the decider passes a message to the MAC module informing it about the dropped frame.

In addition to the signals of interfering frames, a thermal noise signal is applied to any incoming frame's signal. The value for the power of the thermal noise floor was derived from the sensitivity of the used transceiver. Sensitivity is defined as received signal strength, for which – assuming an AWGN channel – the frame error probability (FEP) of a 20 B PSDU is lower than 1 %. As can be seen from Fig. 4.2, this is true for a thermal noise of $N_{\text{therm}} = -100.442\,\text{dBm}$.

The MAC module implements the IEEE 802.15.4 unslotted CSMA/CA algorithm [06], with the slight difference that the status of the channel is only checked once at the end of the CCA period of 128 µs. The implemented MAC module also supports the different CCA modes defined by IEEE 802.15.4-2006. The simulation model above the data link layer is defined by the corresponding protocols and described in the corresponding implementation sections (5.3, 8.2).

4.3 Automated Model Creation

To create a simulation scenario from a real testbed, data about received signal strengths among all nodes of the network has to be collected and evaluated.

4.3.1 Topology Monitor

To collect link quality statistics from a real testbed, I implemented a module TopologyMonitor. Placed on top of a TxPowerLayer module, which uses CometOS' facilities for meta data attachment to influence the transmission power of a frame, it collects

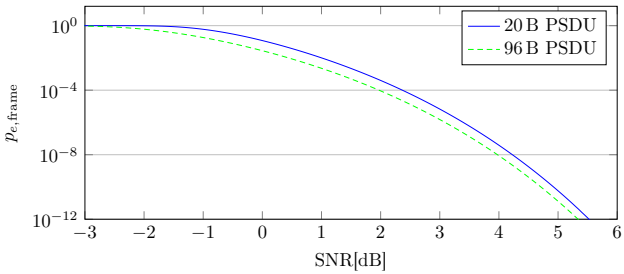


Figure 4.3: FEP ($p_{e,\text{frame}}$) against SINR for different frame sizes

statistics about a node's neighborhood. Two parameters govern its behavior: a period and a frame size. When activated, the TopologyMonitor transmits a broadcast frame with the configured size at a random point in time during each period. The RSSI of incoming frames as reported by the radio driver and its square value are accumulated and stored for each source address, i.e., neighbor. Additionally, the overall count of received frames and the number of occurrences of the minimum RSSI value of −90 dBm (see Sect. 4.2.3) is stored. Each node also keeps track of the number of broadcast frames sent.

The frame size was chosen as 96 B to reflect the size of data frames during a 6LoWPAN fragmentation experiment. Using frames of smaller size slightly underestimates the potential for frame errors (Fig. 4.3). The interval of sending was set to $T_{\text{interval}} = 2\,\text{s}$ to reach a reasonable trade-off between introducing a probability for collisions, which are not desired, and the runtime necessary to get a significant number of samples. The probability for an interval without any collisions under the assumption of independent frame transmissions can be calculated as

$$P_{\text{none}} = \left[1 - (n-1)\,\frac{T_{\text{tx}}}{T_{\text{interval}}}\right]^{n} \tag{4.2}$$

with T_{tx} being transmission duration for a frame and n the number of nodes within interference range. The formula neglects that transmissions are actually not independent due to the CSMA/CA algorithm, that usually not all n nodes will be within interference range of each other and that frames are not always transmitted completely within the interval. For a network of 13 nodes (like the Telematics testbed), for the given values a probability of

$$P_{\text{none}} = \left[1 - (13-1)\,\frac{(96+10+2+4)\times 8\text{bit}}{2\times 250\,\text{kbit}}\right]^{13} \approx 0.75 \tag{4.3}$$

can be estimated as worst case for the probability of no frame collisions within a single broadcast interval.

4.3.2 Post-Processing

The data containing RSSI and the number of frames is used to create a log-normal distribution of incoming RSSI values and to determine a static routing topology.

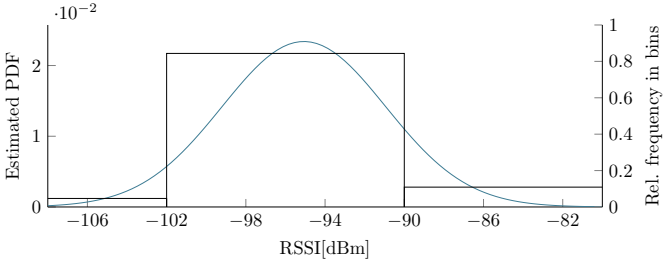


Figure 4.4: Derivation of normal distribution from data collected by the TopologyMonitor; Resulting normal distribution (left y-axis) and relative frequency of samples in bins (right y-axis)

Estimate Normal Distribution

As described in Sect. 4.2.3, the transceiver used for the testbed experiments (Atmel ATmega128RFA1) does report RSSI and ED values with a minimum value of $P_{\min} = -90\,\text{dBm}$. However, average values of $-90\,\text{dBm}$ can be observed for links with an expected transmission count (ETX, [De +03]) of near 1 as well as links with much larger ETX. Given the transceiver's sensitivity of $-100\,\text{dBm}$ this does not come as a surprise.

To get a better estimate of each link in terms of a fixed average plus stochastic log-normal component, the samples of the received signal strength are classified into three bins and for each the number of frames n_{case} is determined (Fig. 4.4):

- "regular" bin (n_{regular}): number of frames with RSSI $> P_{\min}$
- "minimum" bin ($n_{\min}$): all logged minimum values (Sect. 4.3.1)) fall into this category ($P_{\min} \geq \text{RSSI} > P_{\text{loss}}$)
- "loss" bin (n_{loss}): all frames that have not been received are considered to fall into this category (RSSI $< P_{\text{loss}}$)

The boundary for lost frames is set to $P_{\text{loss}} = -102\,\text{dBm}$. These three bins are then used to calculate the parameters of a corresponding normal distribution. Given the cumulative distribution function of the normal distribution

$$F(x) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu}{\sqrt{2\sigma^2}}\right)\right), \tag{4.4}$$

and the number of samples in each bin known, a system of two equations can be created:

$$\begin{aligned} \text{(I)}\quad & F\left(P_{\min}\right) = \frac{n_{\min} + n_{\text{loss}}}{n_{\text{all}}} \\ \text{(II)}\quad & F\left(P_{\text{loss}}\right) = \frac{n_{\text{loss}}}{n_{\text{all}}}, \end{aligned} \tag{4.5}$$

which is equivalent to

$$\begin{aligned} \text{(I)}\ X_m = &\quad \text{erf}^{-1}\left(2\frac{n_{\min}+n_{\text{loss}}}{n_{\text{all}}}-1\right) = &\quad \frac{P_{\min}-\mu}{\sqrt{\sigma^2}} \\ \text{(II)}\ X_l = &\quad \text{erf}^{-1}\left(2\frac{n_{\text{loss}}}{n_{\text{all}}}-1\right) = &\quad \frac{P_{\text{loss}}-\mu}{\sqrt{\sigma^2}}, \end{aligned} \tag{4.6}$$

with $\text{erf}^{-1}(x)$ being the inverse of the erf function. Solving the system for for μ and σ^2 yields

$$\mu = \frac{X_m P_{\text{loss}} - X_l P_{\min}}{X_m - X_l} \tag{4.7}$$

$$\sigma^2 = 0.5\left(\frac{P_{\min}-P_{\text{loss}}}{X_m - X_l}\right)^2 \tag{4.8}$$

Considering all three bins (the "regular" bin is implicitly included by adding to the total number of frames sent), this method yields a unique normal distribution.

A problem with this approach is caused by frames that are lost due to interference, e.g., collisions with other frames. Those may cause a distribution with a large number of frames in the regular bin and a significantly smaller number of frames in the minimum and loss bins, with $n_{\text{loss}} > n_{\min}$. To mitigate this problem, sample mean and unbiased sample variance are directly calculated from all regular samples, ignoring the frames in the other bins, if the number of regular frames is much larger than the number of frames in the other bins. "Much larger" in this context means that the other bins contain less than 0.5 % of the frames in the regular bin. Additionally, in the other case, for each sample in the minimum and loss bins, a value is drawn from the derived normal distribution. These newly created values and the actually measured values from the regular bin are then considered a complete sample. The sample mean and variance of this sample then define the final normal distribution.

The thus calculated distributions with μ_{RSSI} and variance define the physical layer of the simulation model and are used as link properties by the `StaticConnectionManager` and the `EmpiricDecider` (see Sect. 4.2.3).

Static Routing Topology

The link properties as derived in Sect. 4.3.2 are – in combination with the recorded ETX values – also used as input to determine the static routing topology of the simulation model. They form a weighted directed graph representing the network connectivity. To tackle the problem of time-variant links that may cause a static topology to become disconnected (as discussed in Sect. 4.2.3), some adjustments are made to the weights. As basic weight, the measured ETX value is used. Penalties to this basic weight are applied for links with an average RSSI below a certain threshold and links with highly varying RSSI to derive the adjusted weight w:

$$w = \begin{cases} \text{ETX} + k_{\text{abs}} \times (\text{RSSI}_{\text{th}} - \text{RSSI}) + k_{\sigma^2} \times \sigma^2_{\text{RSSI}} & \text{RSSI} < \text{RSSI}_{\text{th}} \\ \text{ETX} + k_{\sigma^2} \times \sigma^2_{\text{RSSI}} & \text{else} \end{cases} \tag{4.9}$$

Hence, links with a good and stable RSSI are preferred to those with a weak and highly varying RSSI, depending on the constant penalty factors k_{abs} and k_{σ^2}. The adjusted weight of the two edges between any two nodes u and v (one for each direction of communication) is then further combined to a bi-directional weight:

$$w_{\text{bi}} = w_{\text{uv}} \times w_{\text{vu}} \tag{4.10}$$

The resulting weight yields the ETX for the unicast transmission of a frame under the assumption that frame and acknowledgment have the same probability of success. While this assumption does not hold true due to the different frame sizes of ACK and data frames, bi-directional (and not asymmetric) links are needed nonetheless. The routing topology is found by applying the Dijkstra algorithm [Dij] to the undirected graph with the bi-directional weights.

4.4 Confidence Intervals

To make significant statements about the results collected from experiments and simulations, multiple runs of all experiment configurations were conducted and confidence intervals calculated for the expectation values of measured quantities. Student's t-distribution was used to calculate confidence intervals according to

$$\left[\bar{x} - t_{(1-\frac{\alpha}{2};n-1)}\frac{S}{\sqrt{n}}, \bar{x} + t_{(1-\frac{\alpha}{2};n-1)}\frac{S}{\sqrt{n}}\right], \tag{4.11}$$

with sample mean $\bar{x}$, the sample variance S^2 and the (one-sided) $(1-\frac{\alpha}{2})$-percentile of the t-distribution with $n-1$ degrees of freedom, $t_{(1-\frac{\alpha}{2};n-1)}$. Unless specified differently, 95 % confidence intervals where calculated, resulting in $\alpha = 0.05$.

5 Basic Forwarding Techniques for 6LoWPAN-Fragmented Datagrams

This chapter introduces a first simulative and experimental evaluation of basic and enhanced route-over forwarding strategies for 6LoWPAN. The results are included in this thesis because they illustrate the necessity of an accurate simulation model as developed in Chapter 4.

5.1 Related Work

Ludovici et al. evaluated different 6LoWPAN forwarding techniques for IPv6 datagrams with and without fragmentation [LCC11]. They analyzed end-to-end delay and packet reception rate (PRR) for a single sender for two route-over and two mesh-under schemes for a line topology of up to five TelosB nodes. They call the modes they define in addition to plain route-over and mesh-under "enhanced route-over" and "controlled mesh-under". The former uses basically the same approach as our Direct (2.2.3): fragments are forwarded immediately and upon reception of the first fragment a virtual circuit is installed, which is used to route subsequent fragments along the same path. The latter mode adds additional checks to the mesh-under forwarding process. If a node receives an out-of-order fragment, it solicits its predecessor for the missing fragment. If the missing fragment is not recovered, the node will stop forwarding any fragments belonging to that datagram. In contrast, their plain mesh-under forwarding also forwards fragments of datagrams that have already been lost.

The results of Ludovici et al. show comparatively high end-to-end latency for the route-over forwarding mode. This is attributed to the reassembly of the whole datagram at every hop. The other three modes perform similar to each other, with plain mesh-under exhibiting slightly higher latency, especially for the shorter paths. Considering the packet reception rates, high losses for larger fragments have been observed. For mesh-under, controlled mesh-under and enhanced route-over, the loss rates increase (non-linearly) from 0 for datagrams of 100 B to over 0.4 for datagrams of 1100 B, with plain mesh-under performing worst. The high loss rates were attributed to collisions on the wireless channel. Considering the short distance of only four hops, with only a single node transmitting, the measured performance appears to be very bad. For the plain route-over mode, the performance in terms of loss rate is significantly better (below 0.04) for "small" datagrams, i.e., those with a payload of less than 900 B. For larger datagrams, buffer drops cause the drop rate to increase to 0.58 for datagrams of 1100 B, i.e., even worse than the mesh-under mode. In general, the results show a need for further investigation of 6LoWPAN fragmentation performance.

Bhunia et al. carried out similar experiments and also analyzed end-to-end delays and loss rates for a small testbed of three nodes. Like Ludovici et al. they used TelosB motes running the TinyOS blip stack [DC12]. Their results are in accord with

those of Ludovici et al. and also show very high loss rates when transmitting large datagrams (> 900 B). A modified route-over scheme, which is claimed to implement a retransmission procedure for missed fragments is shown to slightly outperform the plain modes in the given scenario.

Toscano and Lo Bello evaluated the performance of ZigBee [12b] and 6LoWPAN for low-power industrial scenarios [TB12]. They also used TelosB nodes running TinyOS with the TKN154 [Hau09] to realize a beacon-enabled IEEE 802.15.4 cluster-tree and the blip stack for the 6LoWPAN network, applying low-power listening. Care was taken to tune the duty-cycles of the two different schemes to comparable values and an analysis of how to choose the correct settings for beacon and superframe order for beacon-enabled mode and the active and sleeping periods for 6LoWPAN networks is presented. For low data rate applications they evaluated latency, packet reception rate and update time (the time between packet receptions). From the observed results, a number of conclusions are drawn:

- 6LoWPAN duty cycle depends on the overall network load, making it difficult to plan energy consumption beforehand.
- The IEEE 802.15.4/ZigBee network is able to support lower duty cycles with acceptable performance.
- For higher duty cycles, 6LoWPAN exhibits smaller end-to-end delays and higher reliability.

Wang et al. [Zhu+13] proposed a technique for mesh-under routing in 6LoWPANs, which reassembles packets at some intermediate nodes. Evaluating route-over, mesh-under and their chained mesh-under routing (C-MUR) in a testbed consisting of six nodes arranged in a line topology, they observed that C-MUR achieves a latency between mesh-under and route-over and a better packet reception rate than both for an increasing number of fragments.

A typical issue encountered in multi-hop mesh networks is self-interference, as described in Sect. 2.2.3. This was recognized in the research community of wireless sensor networks. In their publication on the collection tree protocol (CTP), which heavily inspired RPL, Gnawali et al. proposed to decrease the maximum rate at which nodes forward frames via CTP [Gna+09]. They determined the optimal delay experimentally in a small testbed. Optimal in that sense means achieving the best trade-off between reliability and throughput. Based on this investigation, they set the interval a node waits after each transmission to 1.5 to 2.5 times the expected transmission time.

The Flush protocol for bulk data transfer [Kim+07a] also recognizes the need to delay consecutive transmissions along the same path. It installs two mechanisms to prevent self-interference. First, information about transmission time estimates of a node and its successor are piggybacked to every data frame. This information is snooped by preceding nodes on the path and used to adapt their own transmission rate. Secondly, nodes refrain from transmitting with higher rates than any other node along the path. This information is propagated from the destination (root of a routing tree) back to the origin of the transmission.

5.2 Modes

This section introduces the enhancements made to the basic forwarding modes introduced in Sect. 2.2.3. Those are evaluated in addition to the plain Direct and Assembly modes.

5.2.1 Enhanced Direct Modes

With the plain Direct mode, nodes forward an incoming fragment (or add it to a transmission queue) immediately. The transmission rate is thus governed by the rate of incoming frames. This can lead to problematic self-interference between fragments of the same datagram that are sent along the same path, which is typically the case in collection traffic patterns.

To prevent such self-interference, the strategy followed by CTP (see Sect. 5.1) is adopted at the 6LoWPAN layer in two different flavors. The first flavor basically leaves the CTP mechanism unchanged. Observing the mean transmission time for typical 6LoWPAN fragments of 96 B size and a IEEE 802.15.4 minimum backoff exponent of 3, the expected transmission time is set to $T_{\mathrm{tx}} = 6\,\mathrm{ms}$. These 96 B are comprised by 80 B 6LoWPAN payload, 5 B 6LoWPAN fragmentation header and 11 B IEEE 802.15.4 MAC header. After each transmission, a node waits for an interval of

$$1.5T_{\mathrm{tx}} \leq T_{\mathrm{d}} \leq 2.5T_{\mathrm{tx}}, \tag{5.1}$$

before allowing the next transmission. This technique is called direct with Rate Restriction (**Direct-RR** mode).

There are some issues with this strategy that lead to the definition of a second flavor. Depending on the position in the network and the available links to other nodes, as well as the current traffic situation, the transmission time at different nodes or at a single node may vary. Additionally, the overall transmission time does strongly depend on the configuration of the IEEE 802.15.4 link layer. Especially the backoff exponent has a strong influence, because before each transmission at least one backoff is executed. For example, minimum backoff exponents macMinBe of 3 and 5 yield an expectation value for the first backoff of 1.12 ms and 4.96 ms, respectively. Considering a raw transmission time for a 96 B frame of 3.264 ms, the backoff phase does significantly contribute to the overall transmission time. Therefore, I propose a technique that adapts the interval between transmissions according to an estimation of the average transmission time based on measurements. This is implemented by continuously monitoring the transmission times reported by the link layer. The measured value then is smoothed by applying an exponentially weighted moving average (EWMA):

$$T_{\mathrm{tx},i} = \alpha T_{\mathrm{txcurr}} + (1-\alpha)T_{\mathrm{tx},i-1}, \tag{5.2}$$

with T_{txcurr} being the duration of the transmission for the last frame as reported by the MAC layer. The actual delay interval then is again derived from 5.1. This mode is called direct mode with adaptive rate-restriction (**Direct-ARR**). In comparison to the mechanism implemented by Flush, which is based on snooping piggybacked information from the frames of successor nodes, it can be executed with solely local information. Hence, to use Direct-ARR, no changes to the standardized frame format are necessary. The smoothing factor was set to $\alpha = 0.75$ in the 6LoWPAN implementation.

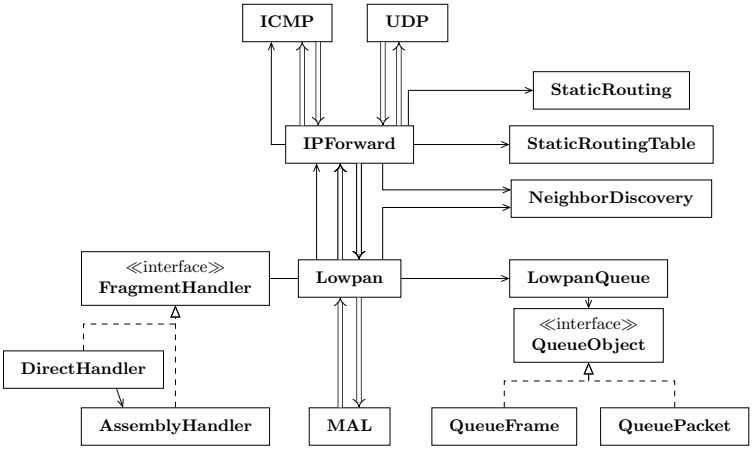


Figure 5.1: High level UML class diagram of 6LoWPAN implementation; double lines represent CometOS message exchange

5.2.2 Retry Control

This chapter also includes an approach from a different angle. With 6LoWPAN fragmentation, the loss of a single fragment results in the loss of the whole datagram, unless some recovery mechanism is applied. Fragments that have been transmitted up to that point have been transmitted in vain and produced network traffic that does not add to the goodput of the network. In lossy wireless networks, link layer retransmissions are desperately needed to prevent high loss rates, which can also be concluded from the results of the analytical model presented in Sect. 3.3. This is taken into account by setting the number of link layer retries to the IEEE 802.15.4 maximum value of 7 (cf. Sect. 5.4). To further prevent the loss of nearly completely transmitted datagrams, the idea of PRC is to increase the number of maximum frame retransmissions the more fragments have been transmitted already. With s_{dg} being the size of the fragmented datagram and $s_{\mathrm{dg,trans}}$ the accumulated size of fragments already transmitted of that datagram, the number of retries r_{PRC} is calculated by

$$r_{\mathrm{PRC}} = 7 + 8 \times \frac{s_{\mathrm{dg,trans}}}{s_{\mathrm{dg}}}. \tag{5.3}$$

This results in a number of 7 to 15 MAC retransmissions, with 15 being the maximum number provided by the transceiver's hardware-supported automatic acknowledgment mode (see Sect. 6.1) used for the testbed experiments described in this chapter.

5.3 6LoWPAN Implementation

Figure 5.1 shows the high-level structure of the 6LoWPAN module for CometOS. For simplification, the result of an additional refactoring for the implementation of LFFR

(cf. Sect. 2.2.5) is not depicted here. The main component is the `Lowpan` module, which interacts via CometOS message passing with the IEEE 802.15.4-like link layer module at one and the IPv6 module at the other side. If `Lowpan` detects a fragmentation header, it passes the incoming fragment to a `FragmentHandler`.

The `FragmentHandler` interface represents the actual and virtual reassembly buffers used for the Assembly and Direct modes of operation (Fig. 5.1). Note that the former is needed in both modes of operation for reassembling datagrams that are destined to a node.

The `AssemblyHandler` delegates the reassembling process to a `DatagramReassembly` object, which manages identification (tag, size, source address), decompression, transit status (bit vector of arrived 8 octet units), timeout information and data storage.

The maximum duration of the (virtual) reassembly process is bounded by a timeout, which is implemented by an aging algorithm. A single timer is initialized to fire every I_{to}. Given a timeout period T_{to}, the creation of a reassembly or virtual buffer forwarding process is initialized with a counter $N_{\mathrm{to}} = \left\lceil \frac{T_{\mathrm{to}}}{I_{\mathrm{to}}} \right\rceil + 1$ and is decremented by one every I_{to}. The counter always reaches zero after a duration in the interval $[T_{\mathrm{to}}, T_{\mathrm{to}} + I_{\mathrm{to}}]$, in which case the timeout is fired and the process aborted. Because only a single timer is needed to implement this strategy (in contrast to one timer for every reassembly process) memory and CPU resources can traded for accuracy of the timeout. Our 6LoWPAN implementation uses a most trivial configuration with $I_{\mathrm{to}} = T_{\mathrm{to}}$.

IPHC decompression and compression are implemented according to RFC6282 [HT11] and are not described in detail in this thesis.

In a similar way, `DirectHandler` uses `PacketInformation` objects that manages the state of an IPv6 datagram in transit. Different from `DatagramReassembly`, it has to additionally store the outgoing tag of the datagram, which is fixed after the `DirectHandler` asked the IPv6 module for a routing decision.

Both fragment handlers use a global 6LoWPAN buffer to store datagrams and fragments. The `AssemblyHandler` always tries to reserve space in the buffer for the whole incoming datagram upon reception of the first fragment. As soon as fragments are transmitted successfully, the buffer space is partially released.

The actual forwarding of fragments is managed by a generic queue, which accepts objects that implement the `QueueObject` interface. This interface contains methods to generate frames, process replies from the link layer and and retrieve some status information. There are two basic types of object representing the different concepts.

- A `QueuePacket` represents a complete datagram. Objects of this type are added to the queue for datagrams originating at a node in both modes of operation and for forwarded datagrams in Assembly mode, i.e., every time a data request is received from the IPv6 module.

- A `QueueFrame` represents a single fragment and is enqueued every time a 6LoWPAN fragment is received from the MAC layer in Direct mode.

The `Lowpan` module queries the queue for the next object every time it is ready to send, i.e., a transmission has finished and the rate restriction mechanism (if used) allows the next transmission. As soon as a link-layer response is received, the result is passed to the corresponding `QueueObject`, which is responsible for updating the transmission

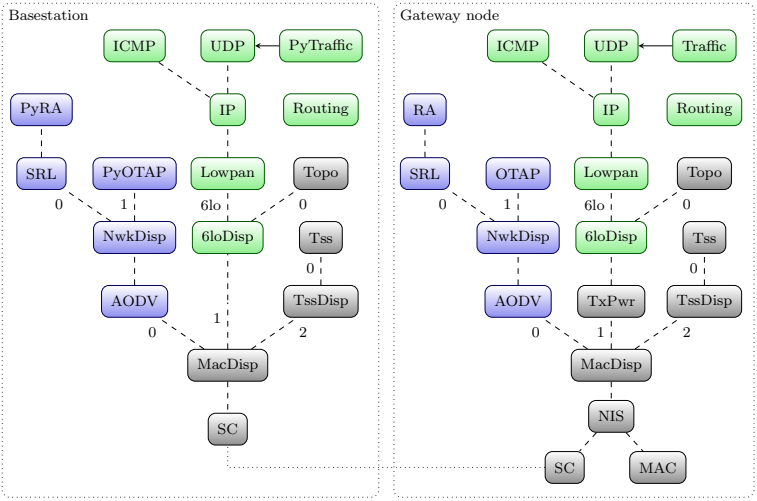


Figure 5.2: CometOS stack used at the at the basestation and the node attached to the basestation, with control stack (blue) and 6LoWPAN stack (green). 'Py'-prefixed modules are specialized basestation modules. Dashed lines represent message-passing connections via CometOS gates. Numbers represent protocol identifiers added to/removed from a frame by the dispatcher module.

status. This comprises checking for completion of the transfer or – in case of failure – to delete all traces of the datagram in the buffer. This implementation implies that datagrams encapsulated in a `QueuePacket` are handled continuously until completion. `QueueFrame`-encapsulated fragments of datagrams in transit may interleave with each other or with a complete `QueuePacket`-encapsulated datagram originating at the node.

5.4 Experiment Setup

This section describes the simulated network topologies and the configuration of the testbed experiments. Furthermore, the mechanism used for time synchronization in the testbed is introduced. The presented evaluation is subdivided into two sets of simulations and experiments, simply named first and second set of experiments. Differences between those two sets are described in the corresponding subsection.

5.4.1 Testbed

For the purpose of validation, we used a testbed consisting of 13 dresden electronic deRFmega128-22A00 modules equipped with an Atmel ATmega128RFA1 controller with integrated IEEE 802.15.4 transceiver operating in the 2.4 GHz ISM band. It provides 128 KiB of program memory and 16 KiB of RAM, which was enough to run

the implementation of the 6LoWPAN stack for CometOS next to a wireless control stack (Fig. 5.2). The actual data traffic was sent through the former, while the control stack was used for topology measuring, OTAP and the control of experiment runs and collection of data. This experiment control is implemented as a set of python scripts, wrapped around a version of the protocol stack compiled for the native platform. This construct is called basestation in the remainder of the dissertation. It makes use of the remote access feature of CometOS to communicate with nodes in the testbed.

The node directly attached to the data sink on the PC runs a slightly different software stack. Instead of being directly attached to the CsmaMac, the MacDispatcher is connected to a so-called NetworkInterfaceSwitch module. This module in turn is connected to a SerialComm and the CsmaMac modules, which provide the same packetized interface to layers above. Depending on the link layer target address, the NetworkInterfaceSwitch chooses one of the two modules (or both in case of a broadcast) to forward the request to. The NetworkInterfaceSwitch was configured to forward all traffic destined to nodes with an IEEE 802.15.4 short address $< 0x100$, which includes the basestation with address 0, to the SerialComm. From the perspective of the protocol stack, this setup is transparent, i.e., the node attached to the basestation acts as if it had a very reliable link towards node 0, without any knowledge that this node is attached via a serial communication.

The CometOS implementation of the link layer interface makes use of the extended operating mode of the transceiver, which means that the CSMA/CA algorithm and automatic acknowledgments and retransmissions are performed by the transceiver. For both sets of experiments, the routing topology was equal to that used in the corresponding simulation scenario. During experiments, a transmission power of 1.8 dBm was used for all frames passing through the TxPowerLayer. Other frames were transmitted using the maximal transmission power of the Atmel ATmega256RFR2 of 3.5 dBm.

While this configuration guarantees smooth operation of the control stack and the time synchronization (cf. Sect. 5.4.1), it causes an inconsistency with respect to transmission power of IEEE 802.15.4 acknowledgments. The radio driver can briefly change the transmission power for outgoing frames, which is realized by attaching some meta-information to the frame. The software acknowledgment layer, however, does not know for which frames it should use a deviating transmission power and therefore transmits all frames with the default, maximal transmission power. Hence, all acknowledgments are sent with a different transmission power than the data frame and thereby have a larger transmission and interference range. With the ACKs being comparatively short and being transmitted shortly after the corresponding data frame, we expected the influence of this anomaly to be of lesser importance.

Time Synchronization

To determine end-to-end latency of UDP packets in the testbed, some time synchronization mechanism is necessary. Therefore, implemented a simple and portable synchronization mechanism for CometOS was implemented. The only assumption it makes about the underlying hardware-dependent radio driver (PHY layer), is that it is able to detect some defined event during the transmission of a frame at transmitter and receiver, e.g., the start or end of a transmission. The radio driver used for the experiments described in this chapter uses the RX_END interrupt, because

accuracy	iMin	iMax	k	noReset
5 ms	50 ms	12	5	false

Table 5.1: Parameters for CometOS simple time synchronization mechanism

a TX_START interrupt is not available with at the Atmel ATmega128RFA1[1]. This point in time is then considered to refer to the same point in real time at both nodes, neglecting the propagation delay of the frame. For the purpose of determining the latency of large datagrams, which is in the order of several tens of milliseconds, a propagation delay of about 1 µs (for 300 m) is considered negligible.

The synchronization mechanism uses a two kinds of broadcasts to propagate a joint network time from a master node to all nodes in the network. Every time update consists of two messages:

- The `InitialMessage` contains only a depth field containing the distance of the sender (in hops) to the master of the time synchronization process. A receiver stores a local timestamp corresponding to a defined event of the reception.

- The `TimestampMessage` contains the network timestamp of the defined event. This timestamp is used by the receiver for synchronization.

Only nodes that consider themselves as synchronized transmit time updates. Hence, time updates originate at the master first and propagate into the network from there. Nodes only process time updates that originate from "parents", i.e., nodes nearer to the sink.

The transmission of time updates is governed by the Trickle algorithm [Lev+11]. Nodes only transmit a timer update if they haven't received k consistent transmissions of other nodes. Consistent in that context means that a time update from a parent does not deviate from the current network time of a node by a configurable accuracy parameter. If, on the other hand, an inconsistent time update is received, the time synchronization mechanism optionally resets Trickle to enable fast propagation of this time update through the network. Because such a reset of Trickle temporarily increases the traffic significantly, an option to prevent resetting was added. Thereby, after a starting phase, time updates are sent using a fixed interval range, reducing the burstiness of the time synchronization traffic at the cost of allowing some inaccuracies in time synchronization.

For the testbed, Trickle was configured according to Tbl. 5.1, yielding a interval range for time updates of $[51.2\,\text{s}, 102.4\,\text{s}]$ in a stable state. The used board (dresden electronic deRFmega128-22A00, [11b]) connects the Atmel ATmega128RFA1 clock input to a crystal oscillator with a maximum deviation of ± 10 ppm and therefore the clock drift between any two nodes in the network should stay below 0.2 ms for an update period. To verify that the time synchronization mechanism did only send a negligible number of frames, the number of transmissions and resets was logged and recorded for each node at the end of each experiment. Thereby, it could be verified that the interval of time updates corresponded to the maximum Trickle interval.

[1] This restriction and the fact that the extended mode of operation is used is the reason for making the time synchronization flexible

Memory usage

For our first set of experiments, the 6LoWPAN layer of our implementation had a 6LoWPAN data buffer of 2000 B. This buffer is used for reassembly, buffering enqueued fragments and as a data storage for datagrams to be transmitted. For the second set of experiments we reduced the buffer size to the IPv6 minimum MTU of 1280 B.

In the Assembly Mode, we set `assembly_entries` to 10, i.e., up to 10 datagrams can be reassembled at the same time (given that their combined size fits into the buffer). Because the Direct modes only reassemble datagrams which are destined to the node itself, we reduce this value to 4 for those. The direct modes use a virtual fragment buffer, which keeps track of the state of up to 15 in-transit datagrams. With this configuration, both modes use about the same amount of RAM yielding a basis for a fair comparison. Concerning program memory, our implementation of the Direct mode uses 3910 B more than the corresponding implementation of the Assembly mode. As the Direct mode basically has to provide the same assembly service for packets destined to the node itself, plus offering the additional services for direct forwarding, this is to be expected. The rate-restricted modes mainly use a single variable of 2 B to store the current estimated transmission time and a CometOS timer object which is started after each transmission corresponding to 5.1.

While the nodes are constrained with regard to the size of the 6LoWPAN data buffer and data structures storing information reassembly or forwarding, for all experiments we provided the base station with buffers and structures large enough to not cause any drops of datagrams. We deem this approach reasonable as we expect 6LoWPAN border routers to be slightly more powerful devices, equipped with larger memories.

5.4.2 Simulation

The configuration of the simulation adopts the methodology presented in Chapter 4, with some exceptions. For our first set of experiments, we used a formula successfully applied in the iEZMesh project (Sect. 2.2.1), which used a sub-GHz transceiver and binary frequency shift keying (BFSK) and consequently calculated the $p_{e,\mathrm{BFSK}}$ by

$$p_{e,\mathrm{BFSK}} = 0.5 \times \operatorname{erfc}\left(\sqrt{\frac{\mathrm{SINR}}{2}}\right). \tag{5.4}$$

This does not accurately represent the direct sequence spread-spectrum (DSSS) technique used at the IEEE 802.15.4 link layer and therefore shows a quite different curve compared to the one obtained by applying 4.1. The mismatch is shown in Fig. 5.3. Using the formula for BFSK, the FEP, denoted by $p_{e,\mathrm{frame}}$, starts to decrease at much larger values for signal to noise ratio (SNR) and the slope of the decrease is less steep. This was accounted for by setting the thermal noise to a lower value of $N_0 = -105\,\mathrm{dBm}$ to get reasonable error probabilities for typical RSSI values between $-85\,\mathrm{dBm}$ and $-95\,\mathrm{dBm}$. While this mitigates the inaccuracy of the model for transmissions in case only the thermal noise has to be considered, in case of interference the model still behaves differently from one using (4.1) to model DSSS. With (5.4), interference between frames most certainly causes frame drops unless one signal is significantly stronger than the other one, i.e., $\mathrm{SINR} > 12\,\mathrm{dB}$. This is different

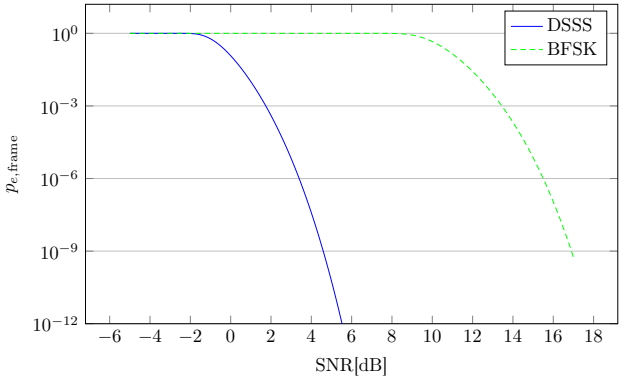


Figure 5.3: FEP ($p_{e,\text{frame}}$) to SNR for BFSK modulation and IEEE 802.15.4-DSSS with O-QPSK for a PSDU of 96 B. For BFSK the FEP curve is shifted and decreasing with a slope less steep.

from the DSSS model, for which interference between frames of slightly different signal strength do not necessarily lead to a loss of both frames. The decision to stick with 5.4 to calculate the FEP proved to introduce an unfavorable bias to the simulation results, which will be discussed in Sect. 5.5.3.

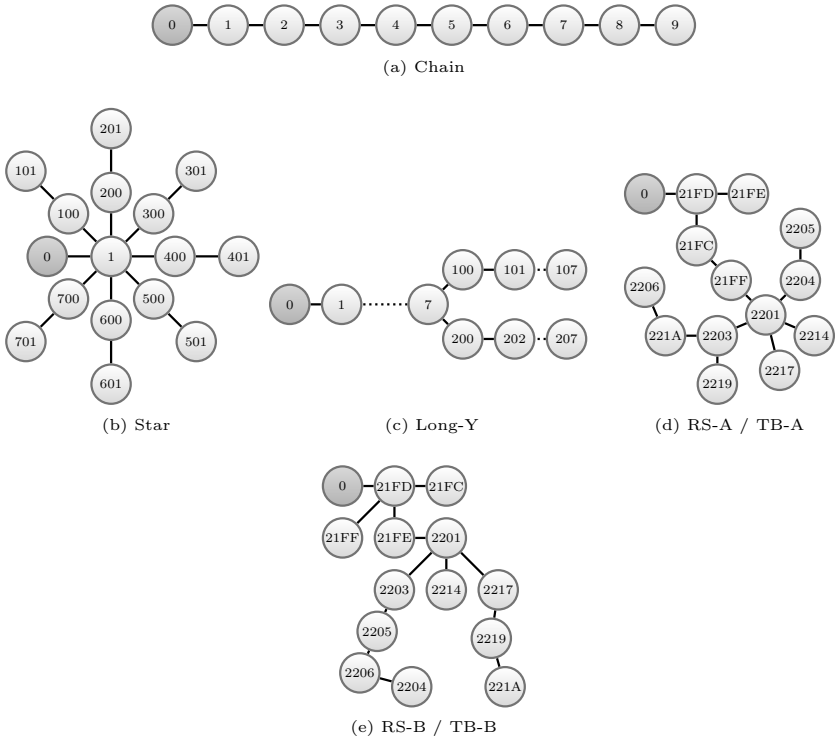
The second set of experiments uses the more accurate formula to calculate the BEP that is provided by the IEEE 802.15.4 standard and a thermal noise as defined in Sect. 4.2.3.

The protocol stack used for the simulations above the MacAbstractionLayer is restricted to the 6LoWPAN stack (cf. Fig. 5.2). The control stack, TxPowerLayer and time synchronization modules are not employed in a simulation environment.

5.4.3 Network Topologies

The evaluation in the first set of experiments concentrated on four different topologies (Fig. 5.4). The first three topologies are idealized, artificial constructs. The “Chain” network (Fig. 5.4a) was chosen as a network with potential for pipelining. In stark contrast, the “Star” network has a maximum diameter of three and therefore does not have any pipelining potential, but employs a bottle neck with regard to contention for the wireless channel and buffer space at its central node (Fig. 5.4b). Finally, the Long-Y network can be seen as a mixture of the two. In those three idealized networks, links are nearly perfect in the absence of interference, i.e., the FEP is near zero. Note that for these networks, the routing tree and the actual wireless connections between the nodes form the same graph. This means that in the Long-Y network, node 100 and 200 are hidden-terminal from each others perspective when transmitting towards node 7.

The two “RealSim” networks were created from data collected in a real testbed with the method described in Sect. 4.2.3. RS-A was evaluated in the first set of



(a) Chain

(b) Star

(c) Long-Y

(d) RS-A / TB-A

(e) RS-B / TB-B

Figure 5.4: Network routing trees for the different topologies. Edges represent static routes, the dark gray node is the sink.

macMinBe (1^{st}|2^{nd})	macMaxBe	macMaxCSMABackoffs	macMaxFrameRetries
3|5	8	5	7

Table 5.2: Configuration of IEEE 802.15.4 link layer; macMinBe was varied between 1^{st} and 2^{nd} set of experiments

experiments, RS-B in the second. Although the real nodes were deployed in the same rooms along a corridor in an office building at Hamburg University of Technology, the time between the first and second set of experiments was several months and the nodes had to be repositioned slightly. Therefore we decided to recreate the (routing) topology for the simulation. The resulting topologies are still similar with regard to average path length (3.62 for the first, 3.92 for the second set of experiments).

Instead of trying to simulate the serial connection in the testbed between basestation and the attached node (see Sect. 5.4.1), in the simulation environment we used a wireless link with a constant strong signal of −50 dBm.

During some preliminary experiments, some links showed a strongly time-variant behavior, i.e., the network became disconnected. To counter that effect, we measured the topology twice, once to determine a routing topology (for testbed and simulation) with lower transmission power of −6.5 dBm and once to collect link statistics for the simulation with a higher one of 1.8 dBm. Thereby, routes are chosen more pessimistically and more distant nodes are more likely to be within transmission range (but also interference range).

5.4.4 Traffic

We evaluated the performance of a collection traffic pattern for all networks. All nodes in the network periodically sent UDP packets of different sizes to the sink. To prevent nodes from transmitting all at the same time, the interval i between two consecutive UDP packet transmissions was randomized:

$$i = I + \frac{1}{2\lambda}, \tag{5.5}$$

with I being uniformly distributed in $\left[0, \frac{1}{\lambda}\right]$ and $\lambda = \frac{\lambda_B}{L_{UDP}}$, where λ_B and L_{UDP} denote the transmission rate of payload and the size of the individual UDP packet, respectively. This ensures a maximum interval between transmissions of UDP packets from an arbitrary node in the network of $i_{max} = \frac{3}{2\lambda}$. This traffic pattern simulates a collection of periodic data from all nodes in the wireless network.

All nodes transmitted a constant total payload of 240 000 B and 48 000 B towards the data sink in simulation and testbed, respectively. Hence, the total number of UDP packets increased and the interval between the transmission of UDP packet decreased with increasing payload size, while the average transmission rate in terms of payload data was constant.

5.4.5 Link Layer Configuration

Because short tests with another macMinBe showed significantly different results for the RS network in simulation, we conducted a second set of experiments for the slightly

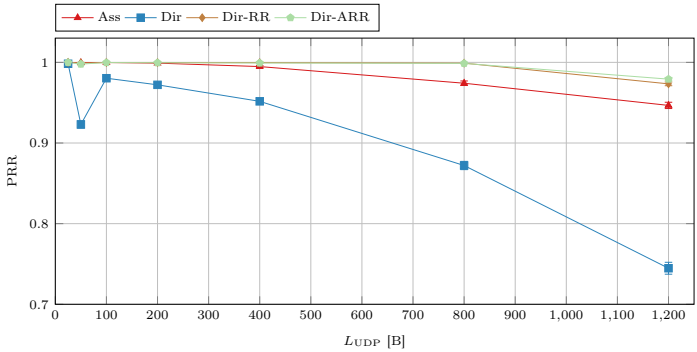


Figure 5.5: PRR of the Chain network for different payload sizes; $\lambda_{\text{B}} = 37.5\,\text{B/s}$

changed RS-B network with a different configuration of the IEEE 802.15.4 MAC layer. The configuration of the IEEE 802.15.4 unslotted MAC is shown in Tbl. 5.2. We set the value for macMaxFrameRetries to the maximum value specified in IEEE 802.15.4.

5.5 Evaluation

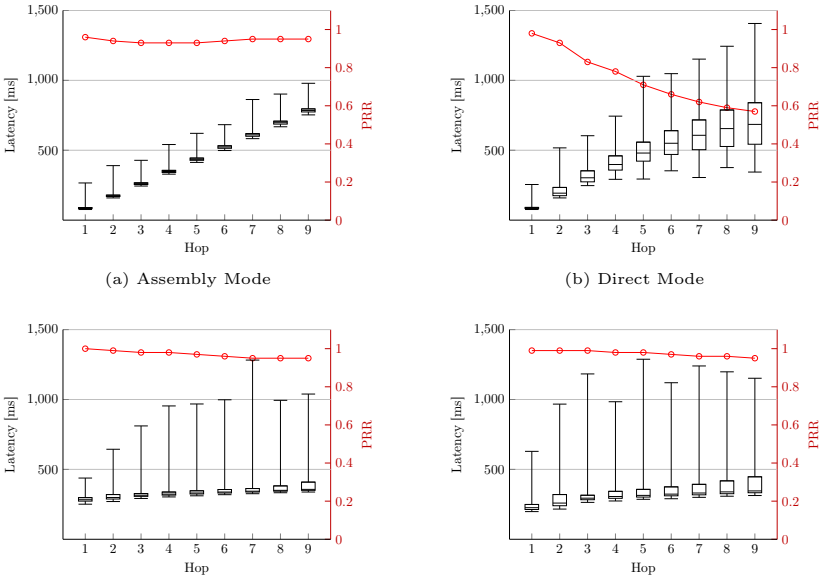
We evaluated PRR and end-to-end latency of the transmitted UDP packets. Latency is shown only for the maximum payload size of 1200 B. For each run, we included a warm-up and cool-down phase of five packets, which were transmitted but not considered for evaluation.

The latency plots show the distribution of latency values against the length of the routing path. Latency values are depicted as boxplots, representing minimum and maximum values by whiskers, 10th and 90th percentiles by the boxes and the median by the middle bar. The latency plots additionally contain an additional y-axis at the right showing the PRR, averaged over all runs and for nodes with the same distance from the sink.

The PRR plots show sample mean of the averaged PRR of all nodes in the network. In the simulation environment, all experiments were repeated at least five times, in the testbed four times. 95 % (two times the standard error) confidence intervals are shown for the PRR plots of the testbed results. Note that in many cases, the confidence intervals for results obtained by simulation were tiny and are sometimes covered by the mark.

5.5.1 First Set of Experiments

For the first set of experiments, we evaluated the Chain, Star, Long-Y and RS-A network topologies.



(a) Assembly Mode

(b) Direct Mode

(c) Direct-RR Mode

(d) Direct-ARR Mode

Figure 5.6: Per hop latency and PRR in the Chain network with $\lambda_B = 37.5\,B/s$ and 1200 B UDP payload; first set of experiments

Chain Network

In the Chain, the rate-restricted modes achieved better PRR – near 1 for packet sizes of up to 800 B – compared to the Assembly mode (Fig. 5.5), while the plain Direct mode heavily suffers from packet losses caused by frame collisions. Main drop reason in Assembly mode is lack of buffer space for reassembly, which is an issue especially for payload sizes ≥ 400 B. The steep drop of the PRR at 50 B payload for the Direct mode is caused by a lack of IPv6 request objects at the 6LoWPAN layer. Those messages are used by the `DirectHandler` to store addresses of fragmented datagrams in transit. Note that a UDP packet of 50 B payload has to be fragmented with our configuration and hence requires setting up transit state, which is not the case for 25 B packets. 100 B packets, on the other hand, are transmitted with a smaller frequency, reducing the required number of IPv6 request objects.

As expected, the direct modes exhibited lower median latency values, especially for nodes more distant to the sink (Fig. 5.6). With the plain Direct mode, the effect is less pronounced and the variation of the latency is considerably larger than for the Assembly mode and the rate-restricted modes. Considering the decrease of the PRR with increasing distance of a node to the sink, the larger and more varying latency in Direct mode can be explained with a comparatively large number of fragment re-

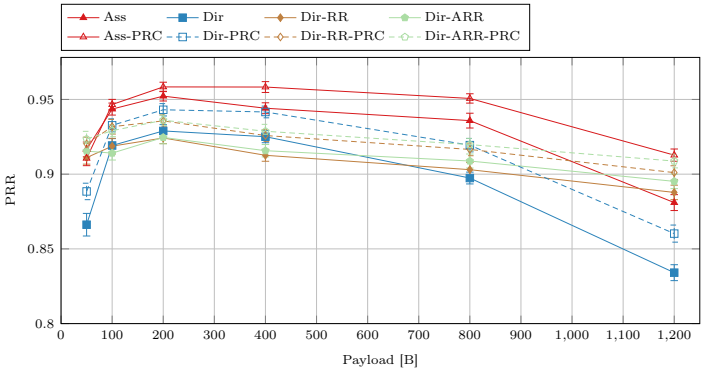


Figure 5.7: PRR of the Star network for different payload sizes; $\lambda_B = 37.5\,\mathrm{B/s}$

transmissions, even for successful UDP packets, which leads to increased transmission times. The rate-restricted modes both show a significantly smaller median latency, with the Direct-ARR mode achieving the smallest. On the other hand, larger maximum latencies can be observed for all direct modes in comparison to the Assembly mode.

Applying PRC does improve the resulting PRR of the plain Direct mode by 0.03, the other modes are unaffected. Hence, result for PRC are omitted for clarity.

Star Network

In the Star, no forwarding mode achieved a PRR of 1. With high potential for collisions at the central node and no potential for pipelining, but also less potential for self-interference, the overall performance of the different forwarding modes was similar, with the Assembly mode showing the best PRR for most payload sizes (Fig. 5.7). We attributed this to the fact that for the most distant nodes, self-interference may occur when the central node is forwarding fragments of datagrams that are still in transmission at the source node. The drop of the PRR in Assembly mode for the largest payload again was caused mainly by lack of available buffer space.

For payloads of 200 B and 400 B, the Direct mode slightly outperformed the rate-restricted modes, which on the other hand achieve a better PRR for larger and smaller payloads. The Star network is the topology that showed the largest effect of the PRC retry control mode: it increased the PRR by 0.02 to 0.04.

Long-Y Network

Similar to the Chain, the two rate-restricted modes performed almost identically (Fig. 5.8). As observed for the other topologies, the Assembly mode performance decreased for large payload sizes, mainly due to buffer drops. For these payloads, Assembly was outperformed by the rate-restricted direct modes. In plain Direct mode, the PRR dropped to values below 0.6.

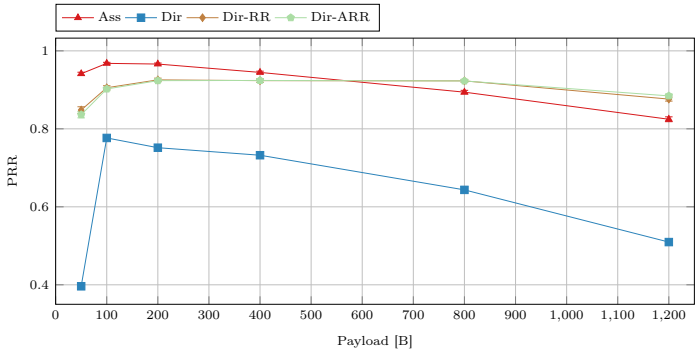
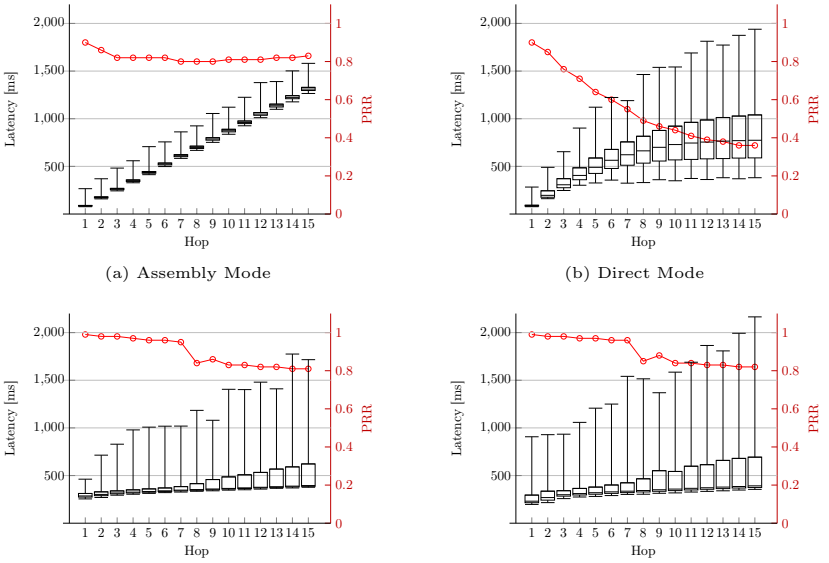


Figure 5.8: PRR of the Long-Y network for different payload sizes; $\lambda_B = 37.5\,B/s$



(a) Assembly Mode

(b) Direct Mode

(c) Direct-RR Mode

(d) Direct-ARR Mode

Figure 5.9: Per hop latency and PRR in the Long-Y network with $\lambda_B = 37.5\,B/s$ and 1200 B payload; first set of experiments

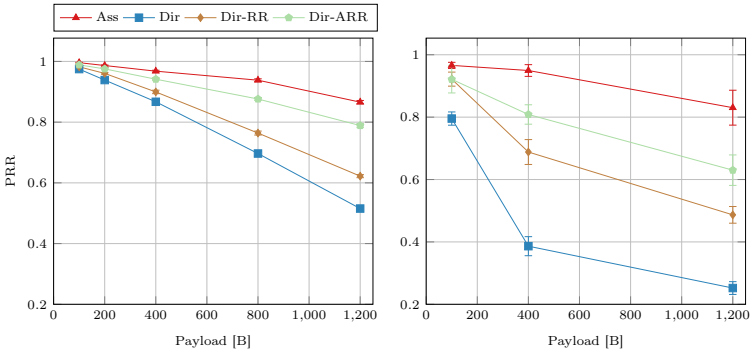


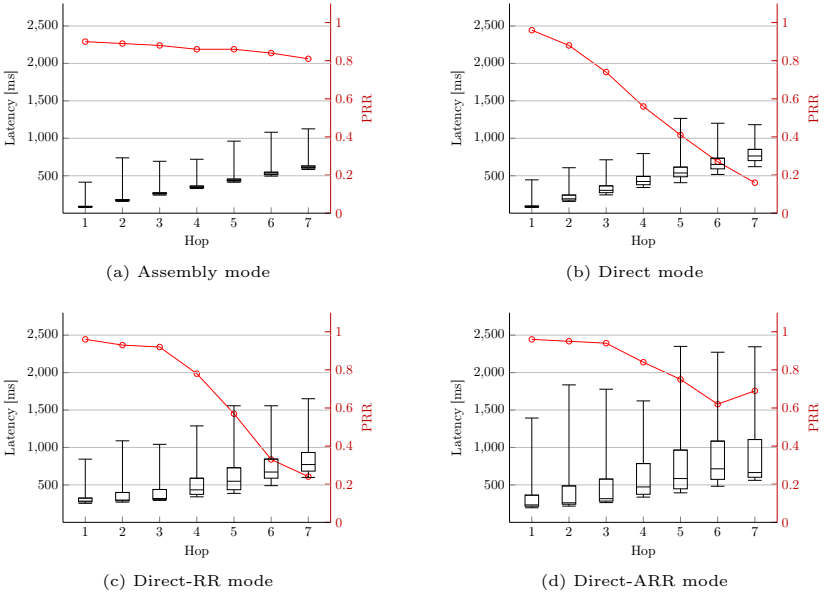
Figure 5.10: Comparing the packet reception rates of the RS-A and the TB-A networks with a byterate of 37.5 B/s; first set of experiments

With the high potential for pipelining of fragments, the rate-restricted modes (Direct-RR: 395 ms, Direct-ARR: 392 ms) achieved median latencies of less than one third of that of the Assembly mode (1311 ms) for the largest payload size (Fig. 5.9). Again this came at the cost of a higher distance between 10th and 90th percentile and larger maximum latencies. Different from the Chain network, the rate-restricted direct modes showed a drop of the PRR beyond the bottleneck node (node 7 in Fig. 5.4c), which also does not occur in Assembly mode.

RS and Testbed

Figure 5.10 shows the PRR of RS-A and TB-A networks, respectively. Payloads of 50 B, 200 B and 800 B were not evaluated in the testbed, but only in the simulation. Compared to to the results obtained from the idealized network topologies, the different direct modes exhibited a significantly worse PRR in relation to the Assembly mode. The difference was even more pronounced for the results from the testbed. In contrast to the results from the idealized network topologies, the Direct-ARR mode clearly outperformed the Direct-RR mode by as much as 0.2.

Considering the results for latency in the RS-A simulation (Fig. 5.11), we observed another difference to the idealized topologies, especially Chain and Long-Y. Although there is some potential for pipelining, the median latency was not smaller but slightly larger with the Direct modes in comparison to the Assembly mode. Also, the PRR dropped steeply for nodes that were four hops or more distant from the sink. For Direct and Direct-RR modes, it dropped below and to values slightly above 0.2, respectively. Only the Direct-ARR mode achieved an average PRR of more than 0.6 for all distances. This indicates that the Direct-ARR mode successfully adapts to additional delays caused by retransmissions or additional CSMA/CA backoffs. The latter can be expected to occur more often in the RS topologies, because the nodes use a higher-than-necessary transmission power resulting in a larger interference range and hence an increased probability for failed clear channel assessments (cf. Sect. 5.4.3).



(a) Assembly mode

(b) Direct mode

(c) Direct-RR mode

(d) Direct-ARR mode

Figure 5.11: Per hop latency and PRR in the RS-A network with 37.5 B/s and 1200 B payload; first set of experiments

Because this also decreases the possibilities for pipelining, it partially explains that the direct modes did not achieve better latencies than the Assembly mode.

5.5.2 Second Set of Experiments

Changing the macMinBe parameter from 3 to 5 was expected to have a significant impact on the overall performance. With macMinBe = 3, the first backoff is chosen from the interval [0 ms, 2.56 ms]. This is even smaller than the raw transmission time for an IEEE 802.15.4 PSDU of 96 B (plus PHY header and SFD) of 3.136 ms and hence causes a high probability of interfering frame transmissions in hidden-terminal scenarios. Note that after a failed transmission, i.e., no acknowledgment was received, IEEE 802.15.4 starts the retransmission resetting the current backoff exponent to macMinBe.

With macMinBe = 5, the initial first backoff phase is in the interval [0, 9.92 ms]. If two nodes start sending at the same time and a frame size of 96 B is assumed, the probability that the two frames do not interfere, is

$$\left[1 - (2-1)\frac{3.072\,\text{ms}}{12.992\,\text{ms}}\right]^2 \approx 0.58. \tag{5.6}$$

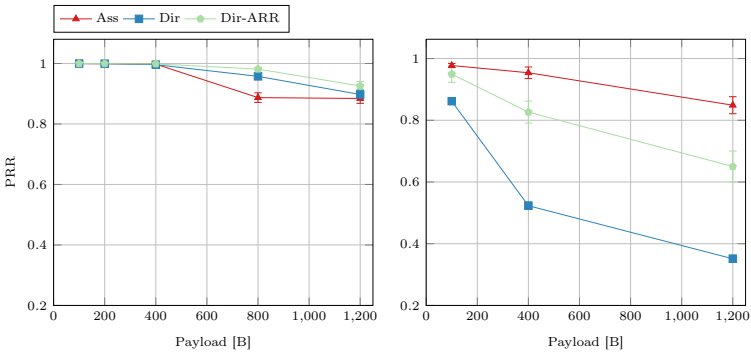
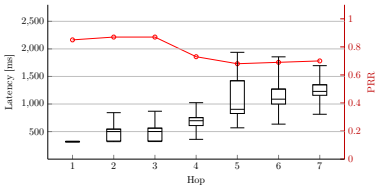
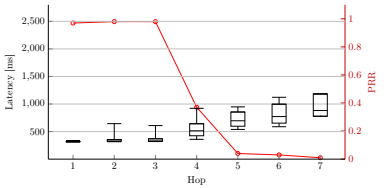


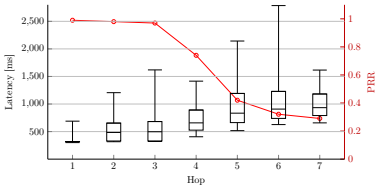
Figure 5.12: Comparing the packet reception rates of the RS-B and the TB-B networks with a byterate of 37.5 B/s; second set of experiments



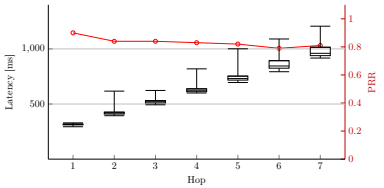
(a) Assembly mode



(b) Direct mode



(c) Direct-ARR mode



(d) Assembly mode (first set)

Figure 5.13: Per hop latency and PRR in the testbed with 37.5 B/s and 1200 B payload; second set of experiments, except Fig. 5.13d, which is a result for macMinBE=3 and is shown for reference (note the different scale on the left y-axis)

Recall also that for the second set of experiments, the BEP for a frame was calculated according to (4.1) in the physical layer model of the simulation.

Figure 5.12 shows the results for the second set of experiments. While the Assembly mode only had a slightly higher PRR compared to the first set of experiments, the Direct and Direct-ARR modes dramatically improved for the largest UDP packets by about 0.4 and 0.13, respectively. Both performed better than the Assembly mode in this regard. The decrease for payloads of 800 B can be explained by the reduced buffer size used during the second set. The configuration allowed only a single 800 B datagram to fit the 6LoWPAN data buffer and therefore increases the number of drops.

Completely different are the results obtained from the testbed. Different from the simulation environment, the only property of the configuration changed was the macMinBe. Here, the Direct mode, though improved by 0.15, did not achieve an even comparable PRR to that of the Direct-ARR mode. Then again, the Direct-ARR mode did not improve in comparison to the first set of experiments and achieves a much lower PRR than the Assembly mode, which showed a very similar performance as in the first set.

The latency plots obtained from the testbed (Fig. 5.13) showed an increase of the median end-to-end latency for the Assembly mode in comparison to the first set of experiments (Fig. 5.13d), which was to be expected with the increased macMinBe. Also, the direct modes achieved better median latency values than the Assembly mode. However, the more striking observation was, that the PRR, while being near 1 for the direct modes at nodes that are not more than three hops away from the the sink, dropped steeply for more distant nodes. For the plain Direct mode, the PRR even approaches 0 for nodes more that 5 hops distant from the sink. For Direct-ARR, the curve had a similar shape, but the decrease did not fall below 0.3 which is still a dramatically bad performance.

A look at the reasons for dropped fragments revealed that for the Assembly mode, drops due to the lack of buffer space in the 6LoWPAN buffer was the primary reason, while for the direct modes, unsuccessful link-layer transmissions caused the majority of packet losses.

The increase of macMinBe in the second set of experiments did not have the expected effect in the testbed experiment. While the latency slightly increased, the overall PRR did not improve by amounts comparable to the simulation environment.

5.5.3 Explanation of Results

At the time this first set of experiments was executed, we tried to explain the observed differences in results between simulation and testbed with the nature of the real world environment:

- People were moving around constantly at daytime during the experiments.
- Numerous IEEE 802.11 hotspots, using the same ISM band may have caused additional interference.
- The transient behavior of wireless links over time may have caused temporal degradation of the link quality between pairs of nodes.

- The mechanism for time synchronization sporadically put an additional small load on the real network.

In combination with the also degrading PRRs in simulation for the first set of experiments, which showed at least similar tendencies to the results from the testbed, the explanation seemed reasonable. The stated reasons, however, fail to explain satisfactorily, why the PRR of the Assembly mode deviates only slightly between simulation and testbed experiments.

Additionally, the PRR dropped to 0.5 and below 0.3 for the Direct mode in the testbed environment in simulation and testbed, respectively. The average byterate of the traffic generator was equal in all experiments ($\lambda_B = 37.5\,B$). Multiplying this value with the average path length in the RS networks results in an (lower bound) estimate for the overall payload data rate of 1.08 kbit/s for the first and 1.176 kbit/s for the second set of experiments. These data rates are not terrifically high in comparison to the IEEE 802.15.4 raw data rate of 250 kbit, even if CSMA/CA, header overhead and additional acknowledgments are considered. This does clearly not represent a congestion scenario. Therefore, the observed performance is unacceptably low.

This reasoning is supported by the fact that changing the BEP calculation in the simulation and using a different macMinBe changed the results, in spite of the smaller 6LoWPAN data buffer, while the testbed results remained virtually unchanged.

Another attempt to explain the results, especially of the second set of experiments, was to make the varying signal strength between retransmissions (drawing a new RSSI from a log-normal distribution; cf. Sect. 4.2.3) responsible for more successful transmissions even in the presence of interference. We deemed this change of the calculation of the BEP to be responsible for the simulation model underestimating the probability of interference between frames leading to corrupted frames at the receiving node(s) and thereby producing unrealistically good results. However, this explanation neglects the fact that in the testbed experiment, the change from macMinBe = 3 to macMinBe = 5 did not lead to significantly different results.

The results clearly imply that the simulation model did not represent the reality satisfactorily. A first attempt to get a better fit involved the introduction of a so-called decider correction factor (DCF). This DCF additionally attenuated the signal of an incoming frame by a certain value if interfering frames were present. Thereby, we wanted to increase the number of frames lost due to interference. This method did indeed decrease the overall PRR slightly, but the results were still completely different from those obtained from the testbed.

I eventually identified the extended operating mode (Sect. 5.4.1) of the transceiver as possible cause for the bad performance observed in the testbed. This mode is activated by putting the transceiver into TX_ARET state, which then automatically performs CSMA/CA backoffs, clear channel assessment and automatic retransmissions. In this state, the transceiver is not able to receive any incoming frames, which means that it becomes "deaf" during the backoff phase. In Chapter 6, I examine the impact of this property and show that it is responsible for the vast majority of frame losses. This realization triggered the conduction of a revisited parameter study, which used the more accurate model for the bit error probability in simulation and a different implementation of the MacAbstractionLayer and is presented in Chapter 7.

In conclusion of the presented results, the major shortcoming of the presented experiment setup was the use of an inadequate formula for the BEP in the simulation

environment (Sect. 5.4.2). It caused interfering frames to be lost with a probability near 1 and – in contrast to the conclusions drawn then – overestimated the impact of interference in the real network. In combination with the unanticipated effects caused by the extended operating mode, simulation and testbed showed strikingly bad but seemingly plausible and consistent results. The execution of additional configurations clearly showed that there existed some mismatch between testbed and simulation model. Therefore, the results presented in this chapter underline how important an accurate simulation model and its validation are.

6 Hardware-Assisted IEEE 802.15.4 Transmissions

In this chapter, the reasons for the difference between results from simulation and testbeds as observed in Sect. 5.5 are examined. The realization of the transmission in hardware, including the CSMA/CA mechanism, is identified as the culprit in this regard.

6.1 Hypothesis

Possible reasons for the pronounced differences between Assembly and Direct modes are discussed in Sect. 5.5. However, these fail to explain, why for the Assembly mode the results from simulation and testbed are very similar, while they differ dramatically for the Direct modes. Especially in the second set of experiments, which uses the formula to calculate the BER as given by the IEEE 802.15.4 standard, the difference is large. One remaining significant distinction between simulation environment and testbed is, that the protocol stack employed in the testbed makes use of Atmel's ATmega256RFR2 transceivers [14] extended operating mode.

This mode provides two additional basic states, RX_AACK and TX_ARET. Entering the former state activates automatic acknowledgments, which are sent autonomously by the transceiver. Its counterpart is the TX_ARET state, which enables automatic retransmissions for unicast frames. With TX_ARET, the transceiver also handles the CSMA/CA protocol of IEEE 802.15.4, i.e., employs a backoff phase and clear channel assessment (and additional backoffs in case of busy channels). A similar automatic mode of operation is found in Microchip's MRF24XA [15]. Such hardware-assisted operating modes are compelling as they can reduce the complexity of the radio driver and link layer and potentially reduce the protocol stack's program and data memory[1], the number of peripherals (timer) and the load on the CPU. Additionally, the processing is faster that way and thereby can speed up the transmission speed of a frame. However, such extended operating modes deactivate the reception of frames in TX_ARET mode. In consequence, no frame can be received during the whole backoff/CCA phase.

In a scenario with multiple hops and 6LoWPAN fragmentation, I expect this "no-RX-while-TX" property of the extended operating mode to be responsible for a large number of frame losses, especially using one of the Direct modes. With multiple frames being available for sending at once (due to fragmentation), on a multi-hop route there is a high probability for consecutive frames to "interfere" in the sense that a subsequent frame is transmitted while it predecessor frame is pending in TX_ARET state at the next-hop node. Figure 6.1 illustrates this situation for an extended

[1] Of the implementations described in Sect. 6.2, the AACK MAC is smaller than the Software MAC by 5280 bytes ROM and 578 bytes RAM

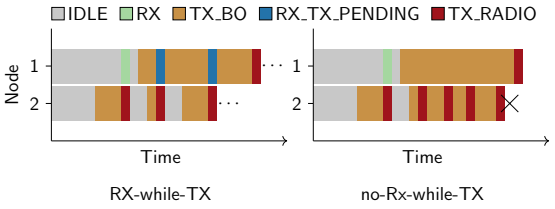


Figure 6.1: Example for sequence of states for two nodes on a multi-hop path leading to the loss of a frame transceivers with "no-RX-while-TX", while "RX-while-TX" transceivers receive all frames

operating mode ("no-RX-while-TX") and a "RX-while-TX" implementation. In the former case, a frame is sent repeatedly into the deaf next hop node, which performs consecutive backoffs and therefore is unable to receive the frame until the sender gives up after a number of retransmissions. Goal of the experiments conducted in this chapter is to verify this assumption.

6.2 Capturing Node State in Real-Time

Three approaches can be applied to evaluate the extended operating modes' impact on the overall performance:

- An analytical approach, based on a model similar to the one proposed by Ludovici et al. [Lud+14], but extended to a multi-hop scenario.
- A simulative approach, using a model which captures the behavior of such an extended operating mode.
- An approach comparing two implementations in a testbed, with the possibility to capture the sequence of states of each node's MAC layer.

I decided to adopt the third approach to be able to eliminate any inaccuracies and limitations introduced by modeling the behavior. As representative for the extended operating mode, I use the MAC layer implementation for the ATmega256RFR2 that is included in CometOS (Sect. 4.1.3, [UWT12]). This implementation is called "AACK MAC" in the remainder of the thesis. As a reference, I ported the radio stack for the ATmega128RFA1 (which is nearly identical to the ATmega256RFR2) of TinyOS to CometOS. The TinyOS radio stack was chosen, because it is widely used and modularized. This layer implements the control of the transmission process comprising acknowledgments, retransmissions, backoffs and clear-channel assessment in software. I also created an alternative backoff layer, which implements the unslotted CSMA/CA of 802.15.4 and replaces the default TinyOS backoff mechanism. This implementation is referred to as Software MAC throughout the thesis.

Keeping track of the accurate sequence of states of each node's MAC layer poses two major difficulties. First, the memory needed to store a large number of state changes is not available on the resource-constrained nodes which already contain a

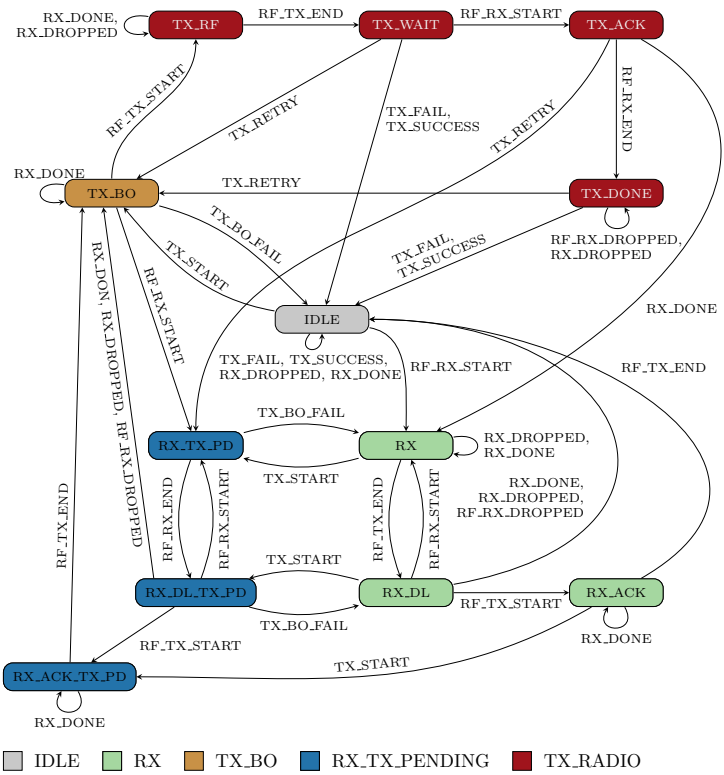


Figure 6.2: State machine for Software MAC

complete IPv6/6LoWPAN stack plus a parallel stack to control the execution of experiments. Secondly, time synchronization with the accuracy of some µs between nodes is necessary to accurately interpret state sequences locally. To achieve such a synchronization, additional frames on the wireless channel are necessary, interfering with the data frames of the experiment. For those reasons, I pursued a different approach and instrumented both MAC layer implementations to encode all relevant events as a 4 bit value and signal them to another microcontroller using plain GPIO ports.

I decided to directly output the present event instead of keeping track of the complete state machine within a node throughout the different layers of the radio stack. This was done in order to make the instrumentation of code as non-intrusive as possible. In consequence, only four CPU cycles (250 ns) are needed to update the value of the GPIO port to signal a new event.

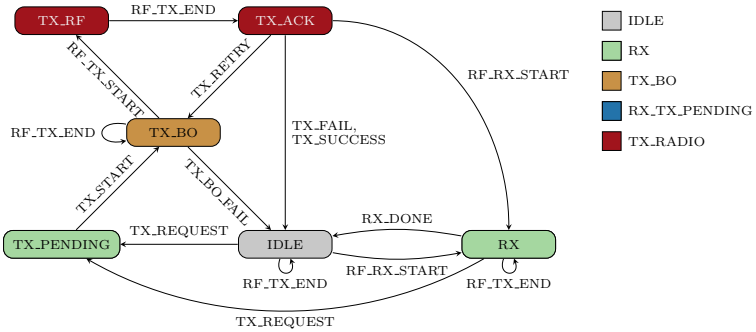


Figure 6.3: State machine for AACK MAC and legend for aggregated states

I took two steps to arrive at a simplified state machine that contains all the relevant information about the MAC layer's state of each node. First, I identified the events and states necessary to unambiguously reconstruct the sequence of states from a sequence of events and created a detailed state machine for each of the two implementations (Figures 6.2 and 6.3). Secondly, several states of the detailed versions were subsumed under a smaller subset of states relevant to the evaluation. Our goal is to especially recognize the occurrences and results of situations, in which a sender transmits, while the destination node is in a backoff phase.

For the Software MAC, I therefore distinguish between "normal" RX states and those RX states, during which the transceiver is processing a transmission request as well (marked by a _TX_PD suffix in Fig. 6.2). The latter are subsumed under an RX_TX_PENDING state. Additional subsumed states for both implementations are IDLE, TX_BO (CSMA backoff), RX (receiving) and TX_RF (sending). Those are represented by different shades/patterns in Fig. 6.2. Note that for the AACK MAC there is no straightforward way to determine if a started reception has been successfully finished. This also includes the reception of frames not destined for this node – in both cases, there is only the absence of an interrupt. Therefore, nodes are often recorded to remain in an "RX" state after an unsuccessful or discarded (filtered) reception, instead of going back to IDLE. As I am exclusively interested in counting "RX-while-TX" occurrences, this does not bias the results in any way.

6.3 Experiment Setup

The actual testbed consisted of four ATmega256RFR2 nodes (labeled 105, 10E, 10D and 10F in Fig. 6.4) in a single large room (about 8×4m), spaced about two to three meters from each other. Customized Cat5 patch cables were used to connect the transceiver's GPIO ports to 16 pins of PORT C of an ARM Cortex-M4 on Freescale's FRDM-K64F evaluation board, as shown in Figures 6.4 and 6.5. The Cortex-M4 executed a simple application with two chained timers, configured to yield a combined timer precision of $266\frac{2}{3}$ ns. This application sampled the state of the 16 input pins in a busy loop and stored every stable (constant for one tick of the timer) change of

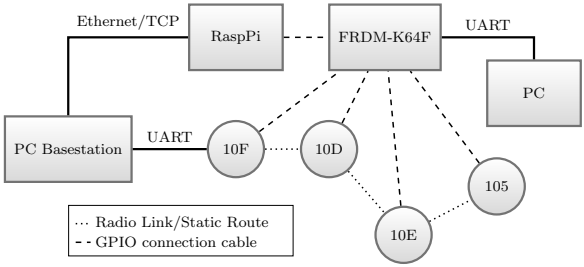


Figure 6.4: Schematic diagram of experimental setup

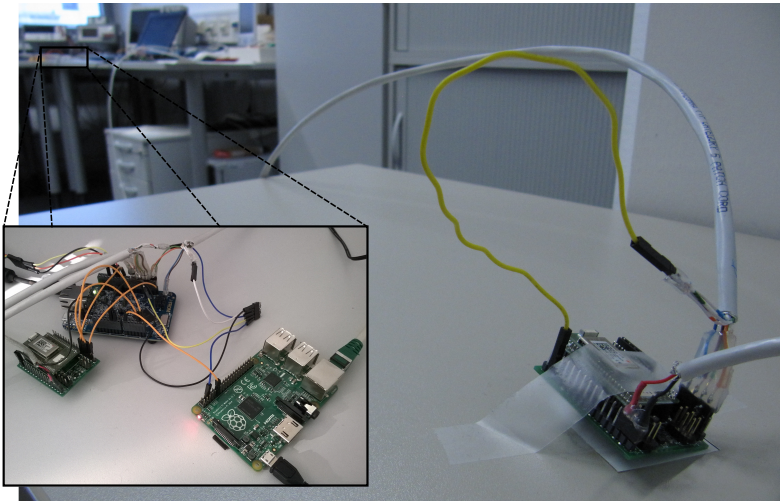


Figure 6.5: Wireless and sampling nodes, and Raspberry Pi controller of experiment

minBe	maxBe	csmaBackoffs	maxFrameRetries	CCA mode	CCA TH
5	8	5	7	0	-90 dBm

Table 6.1: 802.15.4 MAC parameters for all configurations

Mode	Software MAC	AACK MAC
Direct	97 %	21.6 %
Direct-ARR	99.6 %	79.5 %

Table 6.2: Average success rate of datagrams

their value with the corresponding timestamp. The CPU ran at 120 MHz, which was fast enough to sample the input port several times per timer tick. The chosen timer precision, in turn, ensured that no event was missed – the minimum duration between two events was observed to be larger than 4 µs. Upon another GPIO signal, results were sent via UART to a PC and the memory was reset. This signal was generated by the actual base station controlling the traffic generator for the experiment and forwarded via TCP to a Raspberry Pi, which drove the pin.

The nodes used a static routing table to forward the IP datagrams. Only node 105 sent 20 datagrams of 1200 bytes payload to the PC base station. Thereby, additional cross-interference between fragments of datagrams originating at different nodes was eliminated from the experiment. The sending interval was fixed to 4 s, which is more than twice the maximal observed end-to-end delay for a datagram to arrive (or fail). The transmission power of the transceivers was set to the minimal value of −16.5 dBm to realize multiple radio hops between the nodes. Other 802.15.4 MAC parameters were kept constant at the values used in the second set of experiments described in Sect. 5.5 (see Table 6.1). Nodes were configured to use the unslotted CSMA-CA mechanism.

Apart from the Direct mode, I also evaluated the Direct-ARR mode, which improved the reliability for large datagrams (Sect. 5.5). For each MAC layer implementation I carried out experiments using the plain Direct and the Direct-ARR mode, resulting in four different configurations. All experiments were repeated 50 times.

6.4 Evaluation

6.4.1 Direct Mode

The overall success rate for datagrams sent with the Software MAC in Direct mode is dramatically better than that of the AACK MAC (Table 6.2). In comparison to a 97 % success rate with the Software MAC, on average, only 21.6 % of the datagrams reached their destination with the AACK MAC.

The main reason for the observed performance is illustrated in Fig. 6.6. It shows the (aggregated) sequence of states all nodes pass through during a complete transmission of a datagram. As expected, during the transmission of the 18 fragments, a situation occurs in which the next receiver on the path (node 0x10E) enters the TX_ARET state for long enough, that the sending node unsuccessfully tries to send a frame to it

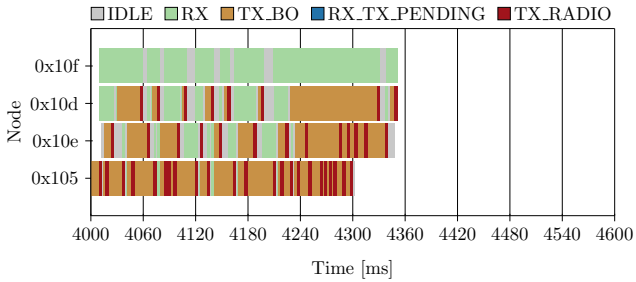


Figure 6.6: Sequence of states; AACK MAC, Direct mode, run 0, datagram 1

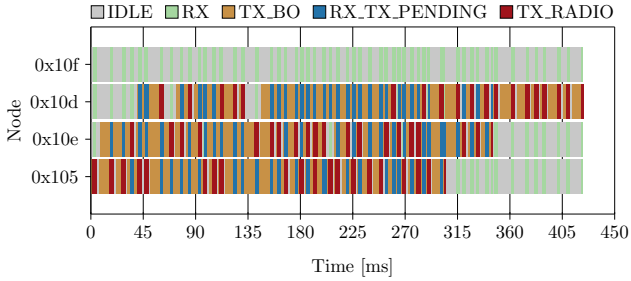


Figure 6.7: Sequence of states; Software MAC, Direct mode, run 0, datagram 0

(recall that during TX_ARET, the transceiver is unable to receive any frame). This pattern can be observed in slight variations for most of the datagrams in all runs for the AACK MAC in Direct mode.

The Software MAC's superior performance can be mainly attributed to the different behavior concerning the reception of frames while being in some TX state (Fig. 6.7). The occasions in which a node received frames while being in a back-off state for it's own transmission are marked as RX_TX_PENDING in the plot. It can be seen that allowing these receptions of frames does not lead to any losses of frames pending for transmission, but on the contrary significantly reduces the number of necessary retransmissions for the sending node.

Departing from individual datagrams to a more general view, the number of fragments in certain combinations of events at the sending node and state at the receiving node were extracted from the experiment data. Table 6.3 shows the summed up and averaged results for the first two nodes of the path (105, 10E).

Comparing these data, it can be observed that the number of total fragments sent (fragRequests) reaches only 57.7 % of the number of fragments needed for a complete transmission (fragRequests (max)) of the datagram, because often senders had to give up the transmission due to a completely failed transmission of a fragment.

		Software MAC		AACK MAC	
(0)	fragRequests (theo. max)	720.00		720.00	
(1)	fragRequests	717.10	99.6 % of (0)	415.30	57.7 % of (0)
(2)	fragSuccessDstTx	518.48		0	
(3)	fragSuccessDstNonTx	198.38		398.76	
(4)	fragFailDstTx	81.96	13.6 % of (7)	423.08	100.0 % of (7)
(5)	fragFailDstNonTx	23.44	10.6 % of (8)	4.44	1.1 % of (8)
(6)	fragFailTotal	105.40	14.7 % of (1)	427.52	102.9 % of (1)
(7)	fragDstTxTotal	600.44		423.08	
(8)	fragDstNonTxTotal	221.82		403.20	

Table 6.3: Average fragment counts of Software MAC and AACK MAC over 50 runs; Direct forwarding mode

Furthermore, there is a difference in the relative and absolute number of retries and transmission failures caused by frames that were transmitted to a sender that was in some TX state (TX_BO or TX_RF), denoted as fragFailDstTx. For the AACK MAC, nearly all (99 %) failed frame transmissions are caused by such frames, compared to 77.8 % for the Software MAC. Also, for the Software MAC, the ratio of failed transmissions with receiver-TX and receiver-non-TX states (fragFailDstNonTx) against their respective totals (fragDstTxTotal, fragDstNonTxTotal) are not far apart from each other: 13.6 % vs. 10.6 %. This suggests that the probability of a successful transmission is only slightly higher if the receiving node is in an idle state for the Software MAC. Possible explanations for this small difference are:

- A Transmission started during a CCA by the receiver is lost with the used implementation (during CCA, SHR detection is disabled).
- Sender and receiver perform their CCA at nearly the same time and both start sending.

Much more pronounced is the overall number of failures, which is only 14.7 % of the total number of transmission requests for the Software MAC, but 102.9 % for the AACK MAC. This means that, using the AACK MAC, on average there is about one retransmission for every initial transmission request.

Finally, a higher percentage of fragFailDstNonTx for the Software MAC over the AACK MAC (10.6 % vs 1.1 % of all fragments with the receiver in a non-TX state) is observable. A possible explanation can be found in the fact that, with the Software MAC, on average more fragments reach the nodes farther down the path and thereby increase the number of collisions due the hidden terminal problem, which is not captured by the selected metrics.

6.4.2 Direct-ARR Mode

Using adaptive rate restriction increases the average datagram success rate of the Software MAC slightly, that of the AACK MAC greatly (Table 6.2).

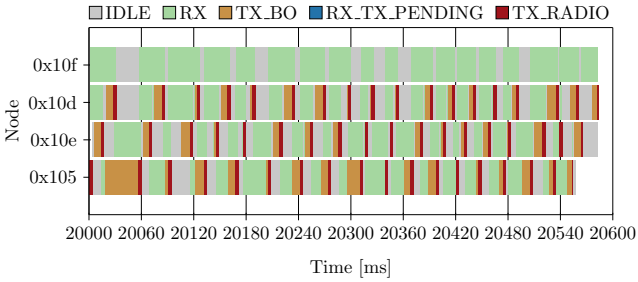


Figure 6.8: Sequence of states; AACK MAC, Direct-ARR mode, run 0, datagram 5

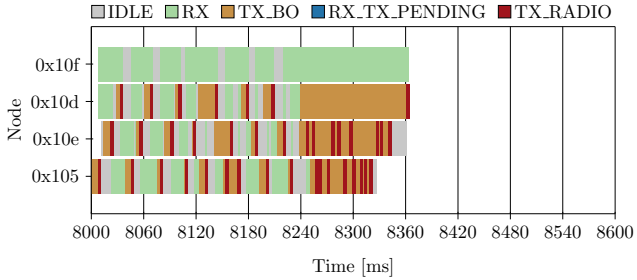


Figure 6.9: Sequence of states; AACK MAC, Direct-ARR mode, run 0, datagram 2

Figures 6.8 and 6.9 show the sequences of states with AACK MAC and Direct-ARR mode for a successful and an unsuccessful datagram transmission, respectively. Figure 6.8 illustrates how the additional delay of the Direct-ARR mode mitigates the risk of deaf receivers. However, due to the inherent random nature of the length of backoffs in 802.15.4 and random failures on the the wireless channel, the delay mechanism does not completely prevent situations, in which again the sender tries to get its fragment to a receiver in TX_ARET state, as shown in Fig. 6.9.

Observing the results of 50 runs, I found that the by far dominating cause for losses and retransmissions for the AACK MAC are again those fragments, which where lost due to the sender being in TX_ARET state (Table 6.4). As for the Direct mode, the number of transmission failures caused by a receiver-TX state is two orders of magnitude larger than that of transmission failures with a receiver-non-TX state.

However, the relative (in comparison to the number of fragments transmissions requested in total) and absolute number of transmission failures is greatly reduced by the Direct-ARR mode, from 427.52 (102.9 %) to 116.5 (17.6 %). Considering the simple traffic scenario with a single sender and only four hops, the performance of the AACK MAC can still be regarded as disastrous, especially compared with the 99.9 % achieved by the Software MAC.

		Software MAC		AACK MAC	
(0)	fragRequests (theo. max)	720.00		720.00	
(1)	fragRequests	719.92	99.99 % of (0)	661.62	91.90 % of (0)
(2)	fragSuccessDstTx	87.80	12.20 % of (1)	0.00	
(3)	fragSuccessDstNonTx	632.10	87.80 % of (1)	657.28	
(4)	fragFailDstTx	16.72	16.00 % of (7)	114.80	100.00 % of (7)
(5)	fragFailDstNonTx	42.44	6.30 % of (8)	1.70	0.26 % of (8)
(6)	fragFailTotal	59.16	8.20 % of (1)	116.50	17.60 % of (1)
(7)	fragDstTxTotal	104.52		114.80	
(8)	fragDstNonTxTotal	674.54		658.98	

Table 6.4: Average fragment counts of Software MAC and AACK MAC over 50 runs; Direct-ARR forwarding mode

	Software MAC		AACK MAC	
	Direct	Direct-ARR	Direct	Direct-ARR
fragRequests	2.3	0.16	16.23	8.84
fragSuccessDstTx	6.4	4.64	na	na
fragSuccessDstNonTx	5.9	4.67	16.75	9.27
fragFailDstTx	3.8	1.93	14.58	6.84
fragFailDstNonTx	2.7	5.88	0.64	0.82

Table 6.5: Confidence intervals of average fragment counts for all configurations

Most clearly, the effect of the rate restriction is shown by rows (7) and (8) of Table 6.4. Apart from preventing hidden-terminal collisions, it also reduces the number of occasions during which senders transmit toward a receiver in TX state. Thereby, the overall ratio of fragments sent during receiver-TX and fragments sent during receiver-non-TX is more than inverted for the Software MAC and significantly changed for the AACK MAC. Interestingly, the absolute number of fragments is nearly the same for both MAC implementations (rows (7) and (8)).

Confidence intervals for all experiments were omitted for clarity from the results tables and are shown in Table 6.5.

6.5 Conclusions

In this chapter, an experimental setup to analyze the sequence of events and states of the MAC layer within a 6LoWPAN network was presented, focusing on the transmission of fragmented large datagrams with the cross-layered Direct and Direct-ARR modes. The results of the experiments in a testbed of four nodes forming a simple line topology have shown, that the extended operating mode of the hardware transceiver is responsible for the bad performance and the observed difference to the results from simulation. Additional mechanisms like rate restriction can significantly improve the performance by preventing situations in which packet losses typically occur, but an

implementation using the extended operating mode is still not competitive and biases experiment results significantly. Moreover, it does so in a rather devious way, which is difficult to distinguish from hidden terminal collisions or random sources of frame loss, especially in cases where the reduction is less dramatic, e.g., for the Direct-ARR mode.

Although only a small testbed was evaluated, it is obvious that the degradation of performance can not be expected to be less severe in larger networks and/or networks with a larger diameter. Application scenarios that involve consecutive transmissions of multiple frames along the same route employing an unslotted CSMA/CA mechanism therefore should avoid using hardware-assisted transmissions in their current state.

This conclusion has triggered a revision of the experiments in Sect. 5.5 using the Software MAC, which is presented in Chapter 7. For the evaluation of the 6LoOF protocol (Chapter 8), I also use the Software MAC implementation in the testbed experiments.

7 Basis Forwarding Techniques Revisited – a Parameter Study

In consequence of the results of the experiments described in Chapter 5 and Chapter 6, a parameter study with a configuration corresponding to the insights gained was conducted. This chapter introduces the new setup and configuration and presents an evaluation for the forwarding modes Assembly, Direct and Direct-ARR. The main goal of the evaluation is to identify a useful configuration of IEEE 802.15.4 parameters for the 6LoOF evaluation.

7.1 Experiment and Simulation Setup

This section describes the configuration of testbed and simulation. I used the collection traffic pattern that was introduced in Sect. 5.4.4. The total traffic amount was set to 120 000 B in simulation and testbed to prevent the simulations, which explored a much larger parameter space, from running for too long due to a large number of runs. The configuration of implementation specific parameters was not changed in comparison to the one presented in Sect. 5.4.1 and hence the memory usage of Assembly and Direct mode was nearly identical.

7.1.1 Testbed

For this parameter study, the Software MAC as described in Sect. 6.2 was employed in the testbed. A new testbed containing 13 nodes in rooms along the corridor on the fourth floor of our office building was installed. In comparison to the network used for the evaluation in Chapter 5, the position of nodes was slightly changed. Fig. 7.1 shows a rough ground plan containing the location of all nodes. The nodes are dresden electronic deRFmega256-23T00 wireless transceiver modules, equipped with an Atmel ATmega256RFR2. Apart from some additional minor functionality of the transceiver

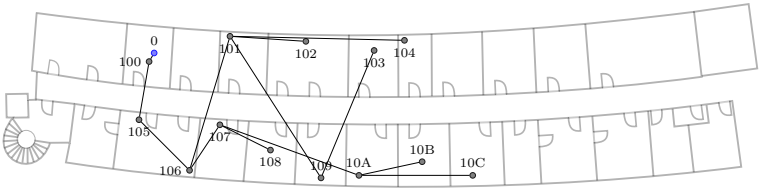
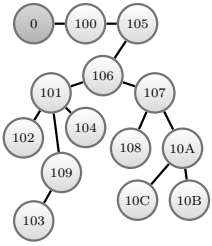


Figure 7.1: Locations of TB-C nodes and routing tree of TB-C

$P_{\text{tx,exp}}$	$P_{\text{tx,topo}}$
6.5 dBm	−0.5 dBm

Table 7.1: Transceiver transmission power for experiment and TopologyMonitor run



(a) RS-C, TB-C

(b) Chain-11

Figure 7.2: Network topologies for the parameter study

and twice as much program and data memory, this controller/transceiver system-on-chip is identical to the Atmel ATmega128RFA1 employed in earlier experiments (Sect. 5.4.1).

The routing topology (Fig. 7.2) of the TB-C network was obtained as described in Sect. 4.2.3. As before, a different transmission power was used for the run of the TopologyMonitor and the actual experiment. These are shown in Table 7.1. All runs in the testbed were repeated five times.

7.1.2 Simulation

The actual parameter study was carried out in the simulation environment, because it took several days to weeks to execute the experiments with a significant number of runs in the testbed. The parameter set employed in the simulations included that of the testbed experiments for validation, but further varies most IEEE 802.15.4 MAC parameters, the size of the 6LoWPAN data buffer and the payload size. I examined the three surviving modes of operation Assembly, Direct and Direct-ARR.

Three network topologies were evaluated: a line topology and two different flavors of the RS-C network (Fig. 7.2). The links in the Chain-11 network are all configured with the same link quality of RSSI = −50 dBm and no variance. The two RS-C networks use the same routing tree but different link configurations. The two configurations use links obtained from the TopologyMonitor with the transmission power of $P_{\text{tx,exp}}$ and $P_{\text{tx,topo}}$, respectively. Hence, one configuration represents the configuration of TB-C, the other the configuration that could be used in the testbed, if links would be less time-varying.

Q	BE_{min}	BE_{max}	macMaxCSMABackoffs	r
1280 B	5	8	5	7

Table 7.2: IEEE 802.15.4 and 6LoWPAN configuration for validation experiments

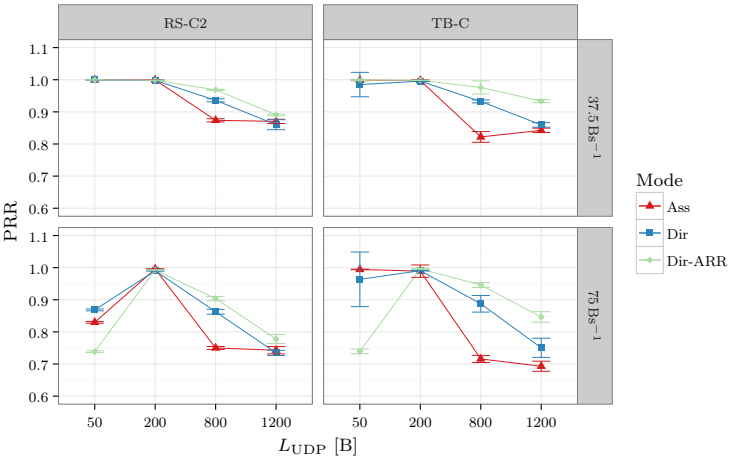


Figure 7.3: Validation experiment; Comparison of average PRR obtained from TB-C and RS-C for two different application data rates. The results between simulation and testbed are largely consistent.

The simulation runs for validation of the testbed were repeated five times. To further decrease the size of the confidence intervals while retaining viable runtimes, this number was increased to 15 for the larger-scale study.

7.2 Validation of RS-C

I used a subset of the parameter space to perform a validation of the simulation model as shown in Table 7.2. The configuration of this validation experiment is shown in Table 7.2 for the 6LoWPAN buffer size $Q$and the IEEE 802.15.4 CSMA/CA configuration values macMinBe (BE_{min}), macMaxBe (BE_{max}) and macMaxCSMABackoffs, as well as the maximum number of retransmissions macMaxFrameRetries (r).

7.2.1 PRR

Figure 7.3 shows the PRR obtained from simulation and testbed runs averaged over all nodes against size of an individual UDP packet L_{UDP} for the three different operating modes and two different (application traffic) byte rates. The same values are shown again in Table 7.3 for exact reference. As a general result, it can be seen that simulation

λ_B [B s^{-1}]	L_{UDP} [B]	Network	PRR		
			Assembly	Direct	Direct-ARR
37.5	800	TB-C	0.822	0.933	0.976
37.5	800	RS-C	0.874	0.936	0.968
37.5	1200	TB-C	0.843	0.859	0.934
37.5	1200	RS-C	0.870	0.860	0.891
75	800	TB-C	0.726	0.888	0.946
75	800	RS-C	0.750	0.863	0.904
75	1200	TB-C	0.693	0.751	0.847
75	1200	RS-C	0.743	0.735	0.778

Table 7.3: Validation experiment; average PRR values for the two largest payload sizes and different total byterate λ_B (rows) from simulation and testbed (columns) with comparable configuration.

and testbed results exhibit similar behavior. For the two largest payloads, the PRR decreases significantly. This effect is more pronounced for the higher application data rate. For the Assembly mode with the two largest UDP payloads, a steep drop of the PRR can be observed from almost 1 to below 0.9 and 0.8 for a data rate of 37.5 B and 75 B, respectively. Both simulation and testbed show the Direct-ARR mode performing best for large datagrams. The plain Direct mode achieves nearly identical values in simulation and testbed, regardless of the configuration setting.

All modes consistently reach a PRR of almost 1 for UDP payloads of 200 B, while the PRR for 50 B packets is significantly lower with the higher data rate in the simulation. At this point, the testbed results deviate from the simulation. The Direct mode shows nearly identical behavior, but Direct-ARR (with a very large confidence interval) and Assembly (reaching a PRR of nearly 1) are different in the testbed. In simulation, the drop is caused by a lack of message structures used by 6LoWPAN and IPv6 layer to communicate, which cannot be observed in the testbed to the same extent, except for the Direct-ARR mode.

For the scenarios evaluated, there is also a consistent bias in the results for the Assembly and the Direct-ARR modes. The Assembly mode achieves an average PRR in simulation which is by 0.027 to 0.052 higher than the corresponding results from the testbed. In contrast, the Direct-ARR mode consistently achieves a 0.012 to 0.069 higher PRR in the testbed. In consequence, the Direct-ARR outperforms the Assembly mode by a significantly larger margin in the testbed for the given settings.

7.2.2 Drop Causes – 6LoWPAN Layer

Analyzing the causes for dropped datagrams shown in Fig. 7.4 for the higher of the evaluated data rates and UDP payloads of 1200 B, it can be observed that general trends again can be observed in testbed and simulation. The vast majority of drops in Assembly mode is caused by a lack of space in the data buffer. Given a 6LoWPAN buffer size of $Q = 1280$ B, which forces the 6LoWPAN module to drop fragments if more than one datagram of those sizes is reassembled, this does not come as a

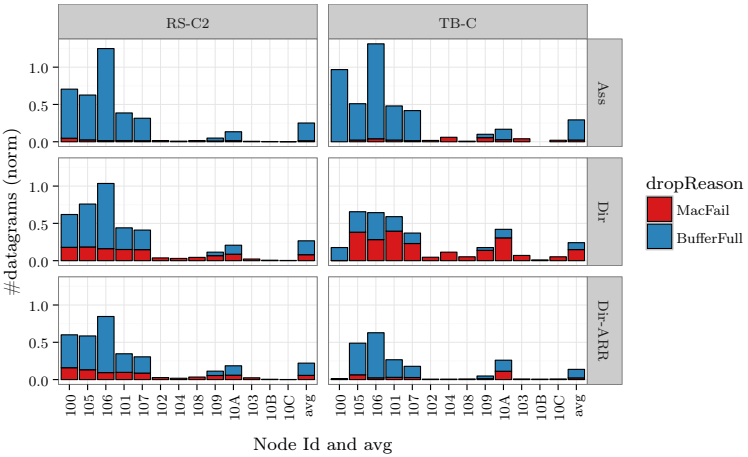


Figure 7.4: Drop causes for each node and average for the whole network, normalized by the total number of UDP packets transmitted per node. Data rate of $\lambda_B = 75\,\mathrm{B\,s^{-1}}$

surprise. It explains the steep drop of the PRR for the Assembly mode. The testbed results show a slightly larger number of datagrams dropped due to failures at the link layer compared to those obtained by simulation. The same can be stated for the Direct mode. While the overall PRR of the Direct mode is nearly identical, the ratio of drops due to link layer failures and due to lack of buffer space is nearly inverse between testbed and simulation.

The Direct-ARR mode shows results different from Assembly and Direct mode: the number of link layer failures and the number of buffer drops in the testbed is smaller than for the corresponding simulation environment.

In addition to these general observations, it can also be seen that the direct neighbor of the basestation drops more datagrams due to link layer failures in the simulation environment than in the testbed. This is a direct cause of modeling this last hop as wireless link, in contrast to the testbed which uses a serial connection that is not governed by the CSMA/CA algorithm and exposed to interference by other node's transmissions.

7.2.3 Drop Causes – Link Layer

A closer look at the causes for drops by the IEEE 802.15.4 link layer reveals that for all examined network topologies a consecutively failed CCA is responsible for the largest share of drops. Figure 7.5 shows the data for the RS-C2 network and the corresponding TB-C. Again, the general trend observed in the simulation is confirmed by the testbed experiment: Direct and Direct-ARR lose considerably more datagrams due to drops at the link layer than Assembly. However, while in the simulated RS-C2

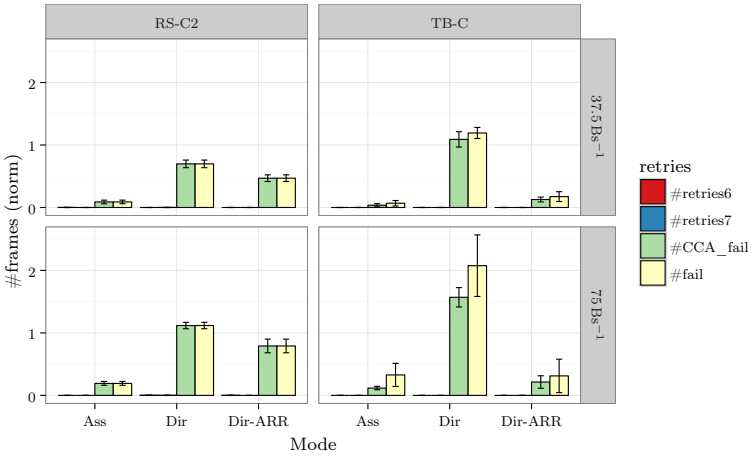


Figure 7.5: Causes of drops at the IEEE 802.15.4 link layer averaged over all nodes, normalized by the total number of UDP packets transmitted per node for simulated and testbed networks (columns) and total byterate λ_B (rows). Additionally the number of frames with a large number of frame retransmissions is shown.

network frames are dropped exclusively due to CSMA/CA failures, in TB-C, frames were also dropped because the maximum number of retransmissions is reached. Note that this can be implicitly deduced from the fact that numCSMAFails is smaller than the numOutgoingNotAcknowledged metric. The number of retransmissions for each frame has not been recorded in the testbed during this evaluation and is depicted as 0 in the plot, while for the simulation environment the depicted 0 represents the actual number of frames with a corresponding number of retransmissions.

The fact that CSMA/CA drops are the sole reason for frame drops in the simulation also explains the surprisingly large number of frames dropped by the link layer at the basestation-attached node: The excellent link quality installed between the two nodes does not save it from CSMA/CA failures.

Additionally, large confidence intervals can be observed for the total number of frames dropped in the testbed experiments for the higher data rate. With the confidence intervals for drops caused by buffer drops being comparatively smaller, it can be deduced that the number of drops due to consecutively failed link layer retransmissions varies strongly between runs.

These results shed some more light on the behavior of the overall PRR between simulation and testbed. For both data rates, the testbed drops significantly more frames after the maximum number of link layer retransmissions. The arguments stated in Sect. 5.5.3 enumerating reasons for frames losses in the testbed, while not logically sound for the results in that section, provide a possible explanation for the effect observed here.

r	$BE_{\min}$	$BE_{\max}$	macCcaMode	Q [B]
3,5,7	3,5,6	7,8	0,3	1280, 2560, 3840
L_{UDP} [B]	λ_{B} [B s^{-1}]	T_{to} [ms]	forwarding modes	
50,200,800,1200	112.5	2000	Assembly, Direct, Direct-ARR	

Table 7.4: Varying and fixed parameters used in the study

For this validation, I used IEEE 802.15.4 CCA mode 0, i.e., the CCA fails when an IEEE 802.15.4-conforming frame is detected **or** an energy detection determines an energy level on the channel that is larger than the macCcaTh, which is set to −90 dBm. The former in both cases is translated to "the CCA fails if a frame is being received". The latter is realized in the simulation environment simply checking the signal strength on the channel once, 128 µs after the CCA has started, instead of performing an energy detection as done at nodes in the testbed. Both Assembly and Direct-ARR mode loose less frames due to CSMA/CA failures in testbed than in simulation – the simulation environment obviously fails more often when executing the CCA in those cases. Contradicting the general trend, for the Direct mode, this number in the testbed is larger. With the available data, I can not offer a satisfying explanation for this result.

7.3 6LoWPAN Forwarding Modes and IEEE 802.15.4 Parameters

In this parameter study, the performance in terms of PRR and latency for three different network topologies is evaluated: a variant of the Chain from Sect. 5.4.3 that contains an additional node (Chain-11) and the two flavors of the RS-C network, RS-C2 and RS-C1. The number of hops the generated UDP packets have to traverse in total are nearly equal: 55 for the Chain-11 network, 57 for the two RS networks. This makes the networks roughly comparable in terms of the total raw traffic load $\lambda_{\text{B,total}}$, which can be defined by

$$h_{\text{total}} = \left(\sum_{l \in \mathcal{L}} \text{len}(l) \right) \tag{7.1}$$

$$\lambda_{\text{B,total}} = h_{\text{total}} \times \lambda_{\text{B}} \tag{7.2}$$

with $\mathcal{L}$ being the set of all routes participating in given collection traffic pattern and len(l) being the number of hops (the length) of a single path.

The overall configuration of the traffic generator is unchanged. It sends UDP packets with an average constant byterate as specified in (5.5).

One goal of the study is to reduce the number of parameters for the evaluation of the 6LoOF study by determining good values for most scenarios that may then remain constant. With buffer drops being the main cause of packet drops in the validation experiments (Sect. 7.2.2) due to a 6LoWPAN data buffer of the size of the IPv6 minimum MTU, the impact of using larger data buffers was also examined. Furthermore, the IEEE 802.15.4 macMinBe ($BE_{\min}$) has shown a high level of influence on the

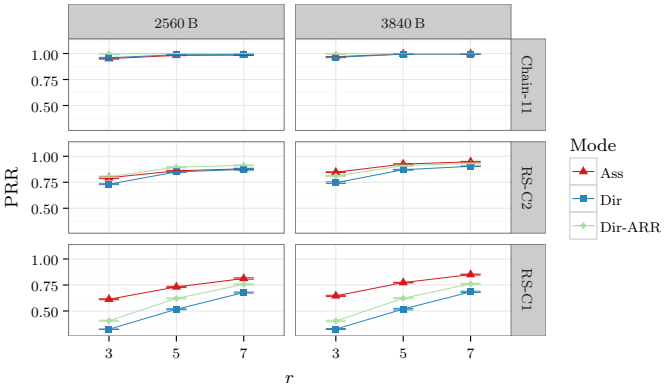


Figure 7.6: PRR against r for different buffer sizes (cols) and networks (rows). For the shown cases, increasing r in all cases increases reliability. UDP payload=1200 B, macCcaMode=3, $BE_{\min}$=5, $BE_{\max}$=8

overall reliability of transmissions in the initial study and is therefore examined for its default value and larger values. To get an indication if the macMaxBe ($BE_{\max}$) has a similar influence, e.g., by shortening the overall transmission duration, a second setting is evaluated in this regard.

The analytical model presented in Sect. 3.2 indicates that a large number of retransmissions successfully prevents rapidly decreasing end-to-end reliability, but is not able to assess the influence of a large number of frame retransmissions on collision probabilities or the utilization of buffer space. The macMaxFrameRetries (r) is varied to evaluate the impact of this parameter.

Although macMaxCSMABackoffs was set to the allowed maximum value according to IEEE 802.15.4, the majority of frames were dropped because of a failed CSMA/CA procedures. Therefore, an alternative IEEE 802.15.4 CCA mode (four different variants are specified) (CCA) was evaluated. The CCA mode 0 was employed in the experiments presented in Chapters 5 and 7. It corresponds to "Detection of a signal with the modulation and spreading characteristics of this standard" OR "Energy above the threshold" of the IEEE 802.15.4 standard [11a]. macCcaMode=3 changes the logical operator between the two conditions to an "AND" and thereby increases the aggressiveness of the CCA, which is expected to lead to less failed CSMA/CA procedures.

An overview of all settings is given in Table 7.4. For each configuration, 15 runs were executed, yielding a total of 19440 (1296 parameter combinations) runs for each network.

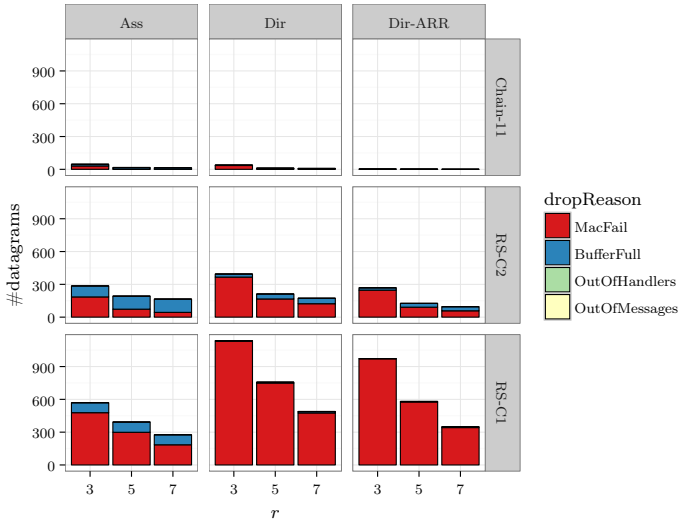


Figure 7.7: Drop causes against r for the different networks and forwarding modes. The number of drops due to lack of buffer space increase slightly for RS networks, but the overall number of drops decreases with increasing r

7.3.1 macMaxFrameRetries

First, the influence of the maximum number of frame retransmissions on the PRR is examined. In all cases, a larger number of maximum frame retransmissions increases the average PRR in most network/parameter combinations. In the worst case, the PRR stays constant, which is the case for the RS-C2 network with macCcaMode 0, because the sole reason for the dropping of frames is a failure of the CSMA/CA mechanism (cf. Sect. 7.2.3). This is exemplarily shown in Fig. 7.6. The Chain-11 network achieves a PRR of almost 1 with the maximum value for $r = 7$ as specified by IEEE 802.15.4. RS-C2 and RS-C1 also benefit from a larger r.

Figure 7.7 additionally shows the drop causes for the same configuration and for a 6LoWPAN buffer size $Q = 2560\,\mathrm{B}$. Although the number of buffer drops slightly increases with increasing r, this effect is more than compensated by a decreasing number of drops at the link layer.

In conclusion, there was no configuration for which allowing more frame retransmissions caused a decrease of the overall PRR. This observation is consistent with the analytical model. Considering that the routes in our setup are static, this result is in line with my expectations. In consequence, the parameter r from is set to $r = 7$ for the remaining evaluation.

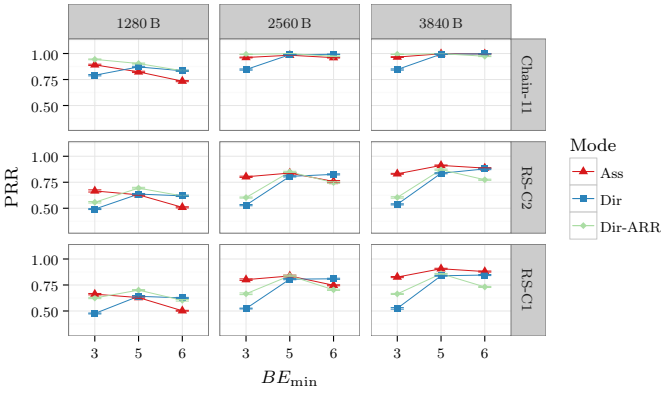


Figure 7.8: PRR against $BE_{\min}$ for different buffer sizes (cols) and networks (rows). $L_{\text{UDP}} = 1200\,\text{B}$

7.3.2 macMinBe

The minimum backoff exponent of the unslotted CSMA/CA of IEEE 802.15.4 is expected to have a large influence of the overall performance. Especially in multi-hop scenarios, a larger setting decreases the probability of self-interference between consecutive 6LoWPAN fragments (cf. Sect. 5.5.2). First, the results are discussed for a $L_{\text{UDP}} = 1200\,\text{B}$, $BE_{\max} = 8$ and macCcaMode = 0.

As can be seen, both Direct modes benefit by the $BE_{\min}$increasing from 3 to 5 (Figure 7.8). In contrast, the Assembly mode, which is much more prone to buffer drops, does only achieve a higher PRR for the larger Q analyzed. With the smallest buffer, the PRR even decreases significantly with increasing $BE_{\min}$. For the given configuration, the Direct-ARR mode shows the best results in the RS networks for $BE_{\min} = 5$. With $BE_{\min} \in [3, 6]$, the PRR is lower. The PRR achieved by plain Direct mode, on the other hand, further increases for $BE_{\min} = 6$ and the effect is more strongly pronounced for larger values of Q.

A larger $BE_{\min}$ increases the probability for the Direct mode to avoid busy channels and hidden-terminal collisions more often, whereas the Direct-ARR mode cannot further profit and even exhibits a decreased PRR. Note that the best PRR achieved by the Direct-ARR mode, however, is in the worst case only slightly lower (RS-C2, largest buffer) and otherwise slightly higher than the best result of the Direct mode.

The causes for drops (Fig. 7.9) reveal the issue that causes the Direct-ARR to lose more packets for $BE_{\min} = 6$. The number of timeouts increases significantly although the number of drops due to other reasons decreases in comparison to the smaller $BE_{\min}$. Note that the number of timeouts is difficult to interpret, because a lost datagram will cause multiple (duplicate) timeout counting events: one at the node where the datagram is initially lost and one at each node that has already received at least one fragment belonging to that datagram. MAC failures and buffer drops may also cause duplicate counting events, but for those cases already forwarded

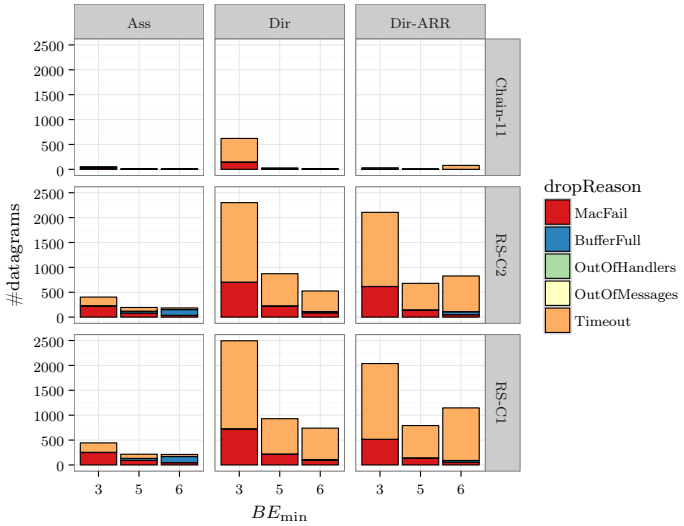


Figure 7.9: Drop causes against $BE_{\min}$ for different networks and forwarding modes. $L_{\text{UDP}} = 1200\,\text{B}$; Direct-ARR is still competitive to Direct, but suffers from the increasing number of timeouts (though no fragment is lost) for $BE_{\min} = 6$

fragments have a good chance to be transported successfully to their final destination (and thereby not trigger another counting event of that kind). However, an increasing number of timeouts and a decreasing number of other drop causes at the same time imply more datagrams have been dropped by a timeout event initially. With the Assembly mode, the ratio of other drop causes to timeouts is about 1; for every dropped datagram, the same datagram times out at the next hop node.

Especially for the idealized Chain-11 network, the Direct-ARR mode achieves a significantly higher PRR for $BE_{\min} = 3$. It can also be seen that this advantage of the Direct-ARR over the Direct mode, melts away with increasing $BE_{\min}$, even when not considering the large number of timeouts. On the other hand, in the comparison of simulation results and testbed experiment the Direct-ARR mode consistently achieved slightly better results (due to less MAC failures) in the testbed, while the Direct mode lost more datagrams to MAC failures.

The PRR achieved by the Assembly mode with the largest examined buffer size (Q=3840 B) is best for $BE_{\min} = 5$, regardless of the network topology. For $BE_{\min} = 3$, especially in the more complex RS networks, MAC failures can be observed, for $BE_{\min} = 6$, buffer drops occur.

With $BE_{\min} = 6$ achieving an only slightly better a PRR only for a small number of cases and causing timeouts of datagrams for the Direct-ARR mode, I settle for $BE_{\min} = 5$ as a compromise for following experiments and the 6LoOF evaluation.

Figures 7.8 and 7.7 also provide some insight into the effect of the 6LoWPAN buffer size. Especially the Assembly mode benefits from larger buffers. Considering that with Assembly, whole datagrams have to be stored at intermediate nodes, this has been expected. The Direct modes lose much less datagrams to buffer drops and hence benefit less strongly from a larger buffer. Although nodes have to put the complete UDP payload into the 6LoWPAN buffer once the traffic generator generates a packet (and thereby also have to store 1200 B), even for the smallest buffer size the available space increases with each transmitted fragment. In general, the Direct modes cope better with smaller buffers.

7.3.3 macCcaMode

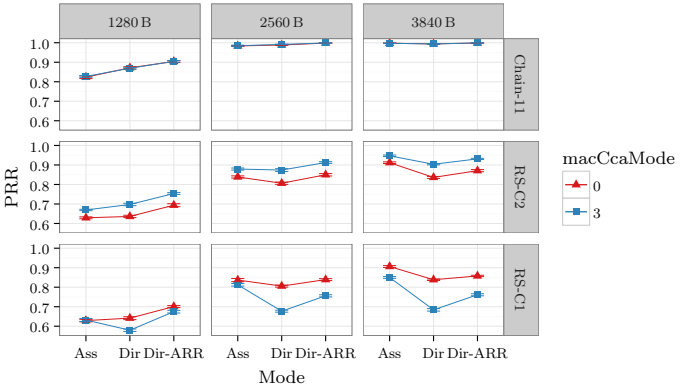


Figure 7.10: PRR against forwarding mode; for different networks and buffer sizes, examining macCcaMode 0 and 3. macCcaMode0 performs better in complex scenarios with many weak links.

The impact of the macCcaMode strongly depends on the network topology under examination. In the idealized Chain-11 network, the transmission and interference range is modeled comparable to a unit disc, where each node is within the communication radius of only the nodes next to it. Hence, the macCcaMode does not significantly change the results for that network. In contrast, the RS-C2 network uses a unnecessarily high transmission power, thereby increasing interference and transmission range far beyond the next hop node. In RS-C1, on the other hand, routes are chosen according to the transmission range.

Figure 7.10 shows the resulting PRR plot. In RS-C2, macCcaMode 3 consistently achieves higher PRRs than macCcaMode 0 for all modes and buffer sizes. In the RS-C1 network, the effect is inverted and macCcaMode 0 performs better.

The reason for the diametrical behavior is shown in Figure 7.11. In conformance with the observations from the validation (cf. Sect. 7.2.3), the dominating cause for the drop of datagrams at the MAC layer are continuously failing CCAs. The MAC

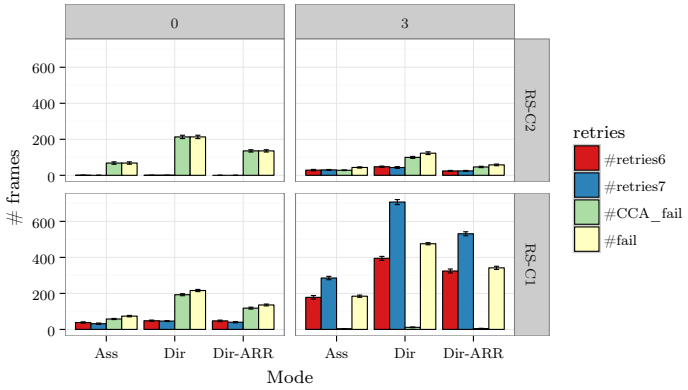


Figure 7.11: Link layer drop causes against forwarding mode; for different RS networks and macCcaMode. The more defensive approach of macCcaMode 0 leads to a large number of CSMA/CA backoff failures, the more aggressive mode 3 to more failures after unsuccessful retransmissions

drops the frame and thereby the datagram after macMaxCSMABackoffs CCA retries. In the RS-C2, the more aggressive macCcaMode 3 significantly reduces the number of such CSMA/CA failures at the expense of a small increase of drops due to failed retransmissions.

For the RS-C1 network, a small share of the drops is caused by failed transmissions with macCcaMode 0. With the more aggressive macCcaMode 3, there is only a negligible number of drops caused by CSMA/CA failures, but a significant increase in drops due to transmission failures can be seen. The latter effect overshadows the former and hence the observed PRR can be explained.

RS-C1 represents the more realistic network topology, because a good routing protocol will always try to find the shortest reliable routes. Therefore, I decide to use the more defensive macCcaMode 0 in following experiments. This is also the mode supported by the radio driver used for the transceivers in the testbed.

The fact that RS-C2 with macCcaMode 3 outperforms RS-C1 with macCcaMode 0 in terms of reliability, does not serve as an argument against mode 0, however. It has to be considered that higher transmission power does improve the reliability in absence of interference and it also decreases the number of potential hidden terminals in this network, because the diameter of the transmission range graph is much smaller than that of routing graph. Therefore, the comparison is inherently unfair. A fair comparison should use routes that have been determined using the same transmission power, which is likely to decrease the resulting diameter of the routing graph as well.

Given the results of the more aggressive macCcaMode 3, it is possible that a lower macCcaTh, e.g., −95 dBm improves the performance. With the Atmel ATmega128RFA1, however, the employed macCcaTh of −90 dBm is the minimal possible value already (cf. Sect. 4.2.3). Therefore, this possibility was not investigated.

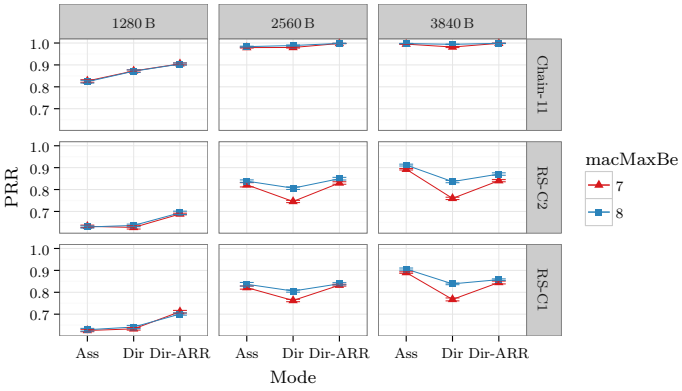


Figure 7.12: PRR against forwarding mode for different networks, Q and $BE_{\max}$. $L_{\mathrm{UDP}} = 1200\,\mathrm{B}$

7.3.4 macMaxBe

With the number of retransmissions set to the maximum specified by IEEE 802.15.4, macMaxBe potentially also has influence on the performance.

Among all examined configurations, none is to be found that increases the reliability for $BE_{\max} = 7$ (Fig. 7.12). The same is true for a macCcaMode of three. The causes for drops (omitted here) show that, while the number of buffer drops slightly increases, this effect is compensated by a more strongly reduced number of drops caused by MAC failures. Hence, I chose the IEEE 802.15.4 default value $BE_{\max} = 8$ as constant value for the evaluation of 6LoOF.

7.3.5 UDP packet size L_{UDP}

Finally, I examine the influence of L_{UDP} on the reliability. With the traffic generation as presented in Sect. 5.4.4, the expectation value of the average (application payload) data rate is independent of the size of the individual UDP packet. Instead, the interval between packet transmissions is adapted. It has to be stressed that the PRR is calculated for the individual UDP packet, there is no relation between them. Therefore, a PRR of 0.9 has a different meaning for small 100 B packets and large 1200 B packets, because in the latter case, all consecutive fragments have to arrive successfully – if exactly every tenth fragment is lost, the PRR would be 0. The comparison may still provides some insight for applications that exhibit a larger number of smaller data blocks, e.g., in a scenario similar to the one described in Sect. 2.5.

Figures 7.13 and 7.14 show the PRR obtained for different UDP packet sizes L_{UDP} for $BE_{\min} = 5$ and $BE_{\min} = 3$, respectively. The results for $L_{\mathrm{UDP}} = 50\,\mathrm{B}$ are omitted, because these exhibit a very low PRR, which is exclusively caused by a too small number of CometOS message structures. It can be seen that with increasing L_{UDP}, the PRR significantly decreases. For $L_{\mathrm{UDP}} = 200\,\mathrm{B}$, it is greater than 0.9 for

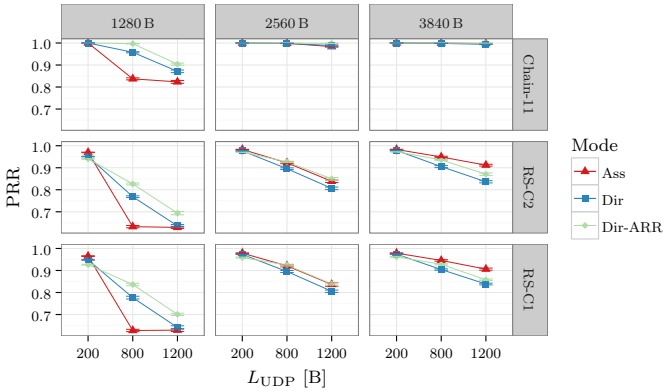


Figure 7.13: PRR against L_{UDP}; different networks, Q and forwarding modes. $BE_{\mathrm{min}} = 5$, $BE_{\mathrm{max}} = 8$, macCcaMode = 0.

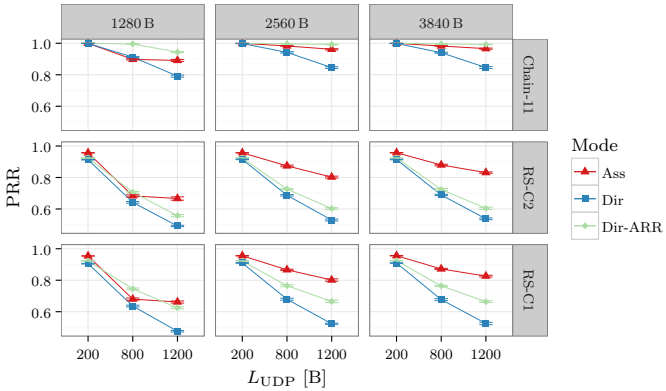


Figure 7.14: PRR against L_{UDP}; different networks, Q and forwarding modes. $BE_{\mathrm{min}} = 3$, $BE_{\mathrm{max}} = 8$, macCcaMode = 0

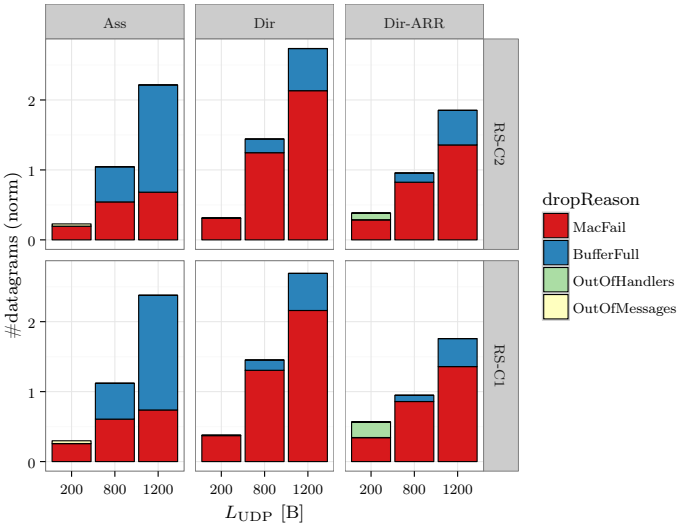
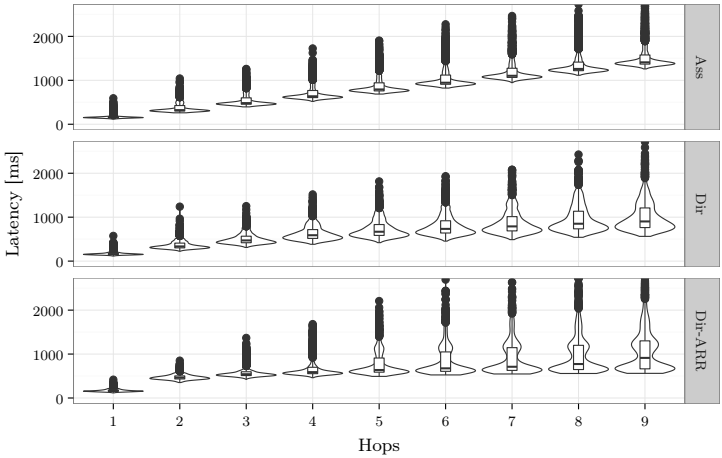


Figure 7.15: Drop causes (total number of packets, normalized by number of packets transmitted per node) against L_{UDP} for different networks and forwarding modes. $Q = 2560\,\text{B}$, $BE_{\min} = 5$, $BE_{\max} = 8$, macCcaMode0

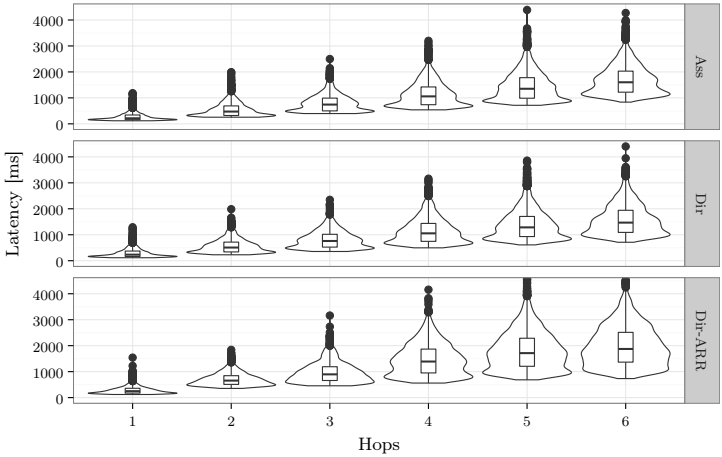
all modes, networks and buffer sizes and both settings of $BE_{\min}$, approaching 1 for most cases.

This indicates that, for the given data rate, it is not advisable to combine smaller packets into larger one, although the reduction of overhead caused by additional IPv6 and UDP headers might be appealing. Because no 6LoWPAN contexts for advanced compression were distributed in the network, only most basic compression is performed and the overhead per UDP packet amounts to 44 B

The causes for drops of datagrams at the 6LoWPAN layer show the reason for the lower PRR achieved for $L_{\text{UDP}} = 200\,\text{B}$ (Fig. 7.15). It is caused by a consistently smaller number of packets dropped due to MAC failures, complemented by a larger number of buffer drops, increasing with the packet size. A small number of 200 B packets get dropped due to missing buffer handlers with the Direct-ARR mode, which are used to manage chunks of memory in the 6LoWPAN data buffer. Their number is governed by the size of a 6LoWPAN fragment in a way that there are enough handlers to fill the buffer. However, as the IPv6 header is stored in a different structure, the first fragment (and the last) use up a buffer handler but not as much buffer space, which explains the observed behavior. Increasing the number of handlers can prevent losses from that source but will consequently lead to `BufferFull` failures.



(a) Chain-11



(b) RS-C1

Figure 7.16: Latency against distance from sink (values accumulated for all nodes with that distance); $BE_{\min} = 5$, $BE_{\max} = 8$, $r = 7$, macCcaMode0, $Q = 2560\,\mathrm{B}$

7.3.6 Latency

I use a combination of violin and boxplot to depict the end-to-end latency. The boxplots show median as horizontal bar and the lower and upper quartile as box. The whiskers represent the minimum or maximum values if these lie within 1.5 times the inter-quartile range; if not, the whiskers represent the most distant measured value lying within 1.5 times the inter-quartile range. Values outside of this range are plotted individually. The violin plots are cut at 1.5 times the inter-quartile range, too.

It can be seen that for the line topology, both Direct modes benefit from pipelining and achieve a lower median latency than the Assembly mode at the most distant node. The upper quartiles of both Direct modes are also lower than the median of the Assembly mode, which increases linearly with the distance. On the other and, the latency varies less with Assembly, even for nodes farther from the sink. Direct and Direct-ARR show similar median values, but the inter-quartile range is larger for Direct-ARR.

For the RS-C1 network, the Assembly mode outperforms the Direct-ARR mode in terms of median latency and again exhibits a smaller inter-quartile range. Latencies of Direct and Assembly perform quite similar in all aspects. For this network topology, the Direct modes clearly cannot utilize the potential for pipelining.

7.3.7 Pull-Based Collection

A most simple alternative to push-based collection is a pull-based approach, i.e., the sink node polls every node in turn for its data. This approach, while arguably not a very sophisticated one, is attractive for a number of reasons:

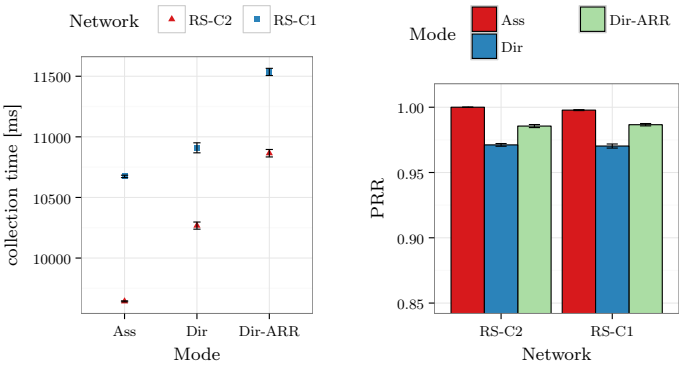
- It is guaranteed that only one traffic flow is active at the same time. Interference does only occur between fragments of the same datagram and this can be prevented by using the Assembly mode.

- The overhead for sending an expectably small (fitting on IEEE 802.15.4 frame) request message becomes relatively small with the size of the data block requested. For example, with $L_{\mathrm{UDP}} = 1200\,\mathrm{B}$ there is one small IEEE 802.15.4 frame for the request and 18 large frames (fragments) for the data.

With a larger network diameter, the potential for parallel transmissions that do not interfere with each other increases, but the overall total throughput is still bounded by one third of the throughput of a single link, because the last hop towards the sink always represents a bottle neck. To achieve a higher throughput, a push-based approach therefore needs to utilize the potential for parallel transmissions without being slowed down by interference between multiple flows.

To be able to quantitatively compare these approaches, additional experiments were run. For those, the basestation/sink requests the transmission of an 1200 B data block from all nodes in a round-robin fashion by means of a very small (2 B) CoAP message. The sink always waits for the current datagram to complete or a timeout of 3 s to occur before it requests the next one. For 100 runs, the average duration of collecting 200 data blocks from all nodes and the total PRR was recorded.

The results show that the Assembly mode achieves the fastest average collection time for the RS networks. A possible explanation can be found in the fact that



(a) Duration for one collection

(b) PRR of data collection

Figure 7.17: Collection duration and PRR for pull-based approach for the two RS networks

the Direct modes have a PRR below one, which means that the sink has to wait for a timeout occasionally. This increases the overall duration. However, though Direct-ARR achieves a better PRR than the Direct, its duration for one collection is larger.

Using the average duration for 200 UDP packets with Assembly in the RS-C1 network as reference (10 673.6 ms) yields an average payload data rate of approx. $112.4\,\mathrm{B\,s^{-1}}$. This rate is all but equal to the data rate used for the push-based collection traffic scenario examined in this chapter ($112.5\,\mathrm{B\,s^{-1}}$). The PRR of the pull-based collection with Assembly is almost 1 and hence it outperforms all RS network configurations of push-based collection examined in this chapter. Even arguing that the used RS networks have a rather small diameter and comparatively mean link characteristics, this has to be considered a rather bleak result.

7.4 Summary

Compared with the experiments presented in Chapter 5, the simulation and testbed setup presented in this chapter produce results that match each other significantly better (cf. Sect. 7.2). Differences can be observed with regard to the causes for dropped frames at the 6LoWPAN layer and at the link layer. However, general trends and effects can be observed in both environments and the results show the same tendencies, albeit not necessarily the same absolute numbers. Therefore, I deem the simulation a viable tool to perform a large-scale parameter study and evaluation of the 6LoOF protocol (Sect. 8.4 and 7.3). Nevertheless, the conduction of testbed experiments to re-validate simulation results is continued for evaluation of 6LoOF.

r	$BE_{\max}$	macCcaMode	$BE_{\min}$
7	8	0	5

Table 7.5: IEEE 802.15.4 MAC configuration parameters found to provide good performance for collection of large, 6LoWPAN-fragmented datagrams

The results of the conducted study have shown that there is a tremendous influence of the configuration of the IEEE 802.15.4 MAC parameters on the reliability that can be achieved in a multi-hop and multi-fragment collection traffic scenario.

For some parameters, it is possible to identify a setting that is beneficial for all or most of the examined scenarios, others were more difficult to assess. macMaxBe and macMaxFrameRetries clearly belong to the former, macCcaMode and macMinBe to the latter category. To prevent parameter explosion (considering that 6LoOF comes with its own set of parameters), I settled for the values shown in Table 7.5 for the following experiments.

Furthermore, it can be stated that the Assembly mode performs well in terms of reliability in the examined RS scenarios. Although being the most straightforward implementation, it is not impacted as strongly by the CSMA/CA or link layer transmission failures the Direct modes suffer from. With enough memory available for a large 6LoWPAN data buffer, it outperforms both Direct modes in terms of reliability, even for $BE_{\min} = 6$.

The behavior is different in the idealized Chain-11, for which the Direct-ARR mode outperforms the other two modes. I assume that it benefits from the narrow interference range, which only reaches to the adjacent node in the line topology, yielding an average degree of two for the graph created by the "within interference range" relation. Hence, it can be assumed that the adaptive rate-restriction is able to prevent interference in many cases. In the RS-C2 and RS-C1 network, the degree of the link/interference graph is 11.85 and 11.15, respectively. This leads to more failed CCAs and retransmissions and thereby increases the variance of the average transmission duration T_{tx} . Due to the EWMA smoothing, I expect the rate-restriction to work less effectively if the variance of T_{tx} is large.

Examining a most simple pull-based collection approach, the superiority of this method compared a push-based collection with randomized transmission intervals has been discovered for the examined network topologies and UDP packet sizes of 1200 B. In consequence, the evaluation of the 6LoOF protocol also considers data rates with the potential for higher throughput than the corresponding pull-based collection approach.

8 6LoWPAN Ordered Forwarding - 6LoOF

This chapter introduces 6LoWPAN ordered forwarding, which was created as part of this doctoral thesis. The 6LoOF mechanism's goal is to avoid the typical problems encountered in the presence of 6LoWPAN fragmentation, as described in Chapter 7.

8.1 The 6LoOF Mechanism

All forwarding strategies presented in Sect. 7.3 (Assembly, Direct and Direct-ARR) exhibit packet reception rates significantly less than 100 %. The main reasons for the packet drops are full forwarding buffers and consecutively failed clear channel assessments on the wireless channel in Assembly and Direct modes, respectively. I assume that these losses in the Direct modes mainly occurred in situations in which multiple nodes try to forward large, fragmented datagrams within the same neighborhood, which resulted in increased channel contention and increased traffic volume, i.e., a local congestion. In consequence, an increased number of fragments and thereby datagrams were lost at the link layer in spite of repeated retransmissions.

6LoOF is an extension for direct forwarding that aims at reducing the number of concurrent transmissions in the immediate neighborhood of nodes. This is realized by utilizing the information provided by the 6LoWPAN fragmentation mechanism and adapting the rate at which nodes transmit their 6LoWPAN fragments. Simplistically speaking, nodes that overhear their current next hop forwarding fragments belonging to a different datagram will completely stop the transmission of their current datagram to allow for faster forwarding and less contention. As soon as transmission of the "foreign" datagram succeeded, the suspended transmission is reactivated. This mechanism implements a congestion control with a narrow spatial and temporal scope. As it does not directly influence end-to-end sending rates, it does not prevent network congestion when the overall offered traffic is above the capacity of the network. I expect higher layers (TCP, CoAP) to deal with end-to-end congestion control.

This treatment of datagrams as traffic flows that suspend other traffic flows implies that nodes do not forward individual 6LoWPAN fragments using a FIFO approach as in the straightforward direct forwarding strategies. Instead, nodes always forward whole datagrams, i.e., they **order** incoming fragments by their belonging to a certain datagram. Thus, the terms *current next hop* and *active datagram* here and in the remainder of the chapter describe the link layer destination of the datagram that is currently forwarded and this datagram itself, respectively. Note that with a route-over approach, even using direct forwarding (cf. Sect. 2.2.3), this next hop does not change during transmission of a single datagram.

8.1.1 Snooping

6LoOF heavily relies on snooping of 6LoWPAN frames prefixed by a fragmentation header to (re-) identify datagrams sent by other IPv6 routers. A node especially needs to identify transmissions of its current datagram originating at the current next hop for this datagram. Using route-over forwarding, the `datagram_tag` changes with every wireless hop. In consequence it is necessary to combine IPv6 header information and 6LoWPAN fragmentation header to achieve a reasonably accurate identification of datagrams. 6LoOF implements two different checks when it overhears a transmission of its current next hop.

First, snooping the first fragment of a datagram, IPv6 header fields source and destination are used for comparison. Note that the basic IPv6 header always fits the first fragment (cf. Sect. 2.2.1). Using the IPv6 header, the node can use destination and source address along with the size of the datagram to check if the transmitted fragment is part of the node's active datagram and thereby can decide whether to succeed with its transmission or suspend it. The corresponding 6LoWPAN `datagram_tag` is stored as metadata for the active datagram and is used to check further overheard fragments.

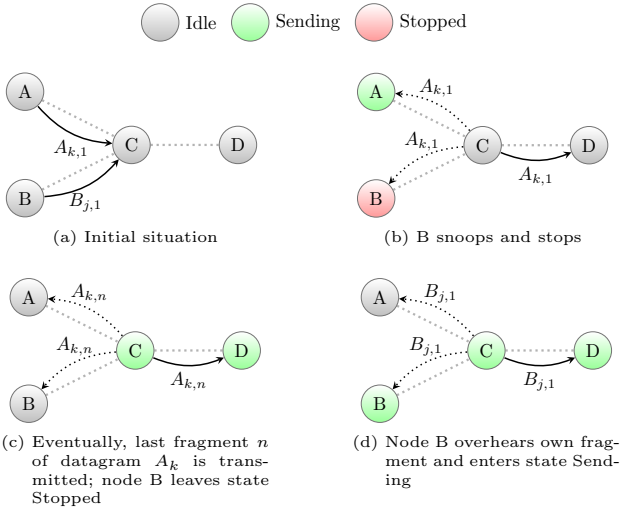
Secondly, if a FRAG_N header is decoded without having a tag stored for the current datagram, 6LoOF resorts to a sanity check. The fragment header's size and offset field are checked to determine if the overheard fragment may be part of the active datagram, i.e., sizes have to be equal and the overheard offset smaller than the current one for the fragment. Note that in case of false positives, this makes 6LoOF behave similar to the corresponding non-6LoOF forwarding mode with regard to the rate of transmissions.

For snooping to work, it has to be assumed that wireless links between nodes are symmetric. All IEEE 802.15.4 unicast data traffic uses link-layer ACKs to confirm reception of a transmission, i.e., transmissions are only considered successful if a matching link-layer ACK is received. Routing protocols used above a IEEE 802.15.4 link layer also have to consider that fact. Therefore, the assumption that wireless links are symmetric between adjacent nodes in a routing graph is reasonable.

8.1.2 Probing

The general concept is illustrated in Fig. 8.1. Here, nodes A and B use node C as their next hop. Node C decides to forward the fragments of node A. This is overheard by nodes A and B (Fig. 8.1b). The former recognizes its active datagram and continues transmitting fragments, the latter recognizes a different datagram and stops sending (Fig. 8.1c). Only after node B overhears the last fragment of A's datagram (Fig. 8.1c), it enters state Sending again (Fig. 8.1d).

A problem with using a snooping-based approach is that in the absence of overheard frames, a node is not able to know if this absence is due to its next hop currently suspending its transmission or simply due to a random loss of frames on the wireless channel. In the latter case, nodes should continue transmitting to get a chance to snoop from their successor. In the former case, however, it is not advisable to continue sending fragments to such a suspended node, as transmissions are likely to cause hidden-terminal collisions at the suspended node. Figure 8.2 illustrates this issue with node D being the suspended node. Furthermore, ACK frames sent from D to C



(a) Initial situation

(b) B snoops and stops

(c) Eventually, last fragment n of datagram A_k is transmitted; node B leaves state Stopped

(d) Node B overhears own fragment and enters state Sending

Figure 8.1: Illustration of the general concept behind 6LoOF

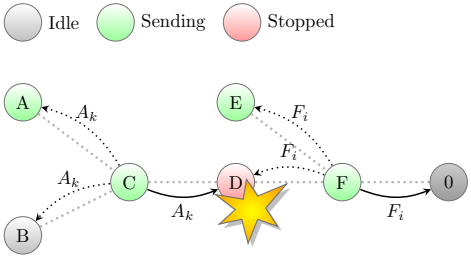


Figure 8.2: Potential hidden terminal collisions at a suspended node D

Dispatch	Header
11 010xxx	FRAG1_EPN
11 110xxx	FRAG1_EPN

Table 8.1: Proposed header dispatch types to signal a node's probing state (explicit probing notification)

may also cause collisions at F with the traffic flow from E to F. In situations, in which there is more than a single sender, the suspended node's buffer also has to store the incoming data without being able to empty it, up to a point at which the datagram has to be dropped.

To mitigate this problem, a probing state for 6LoOF nodes is introduced. Nodes that do not snoop any fragment originating at their current next hop after a defined number of sent fragments enter this state. Similar to the CSMA/CA protocol of IEEE 802.15.4, probing nodes exponentially reduce the frequency of their transmissions until they are able to snoop a fragment from the current next hop. Thereby, contention on the wireless channel is reduced if a confirming fragment was lost or the current next hop is suspended. Considering that the loss may also have been caused by a collision on the wireless channel, reducing the rate of transmissions is a reasonable measure to take for both cases.

Additionally, nodes forwarding towards a probing node have no possibility to know that this node is probing. In consequence, such nodes will snoop a "probing" fragment and if it matches their own datagram, will continue transmitting at a normal rate, governed solely by the rate restriction mechanism as described in Sect. 5.2.1. To prevent nodes from transmitting to probing nodes at normal rate, additional header types for 6LoWPAN can be specified. These use the same format as FRAG_1 and FRAG_N header of RFC 4944 ([Mon+07]), but a different dispatch to signal the probing state of a node. The dispatch values are chosen from the reserved range of dispatch values and do not interfere with any currently used dispatch values. They are shown in Table 8.1.

With the first draft of 6LoOF, preventing probing nodes to receive data at maximum rate was considered important enough to to justify an alteration of the 6LoWPAN standard headers. Reviewing this issue, it was concluded that sending to a probing node at maximum rate is less harmful than thought initially. Given a routing topology that tries to prevent unnecessarily long routes, a node that sends towards a probing node is usually two hops away from the node in state Stopped that causes the node to probe. While this is by no means a guarantee that the stopped node is outside the interference range of the sending node, there is a good probability that the strength of potentially interfering signal is much lower than that of the fragments the stopped node is trying to snoop. Therefore, the usage of these header dispatch types is optional for 6LoOF and is defined by the boolean parameter useEpnFlag x_{EPN}. The impact of this parameter is examined in Sect. 8.4.1.

8.1.3 6LoOF Definition

Figure 8.3 on the following page shows a UML statechart defining the core of the 6LoOF protocol.

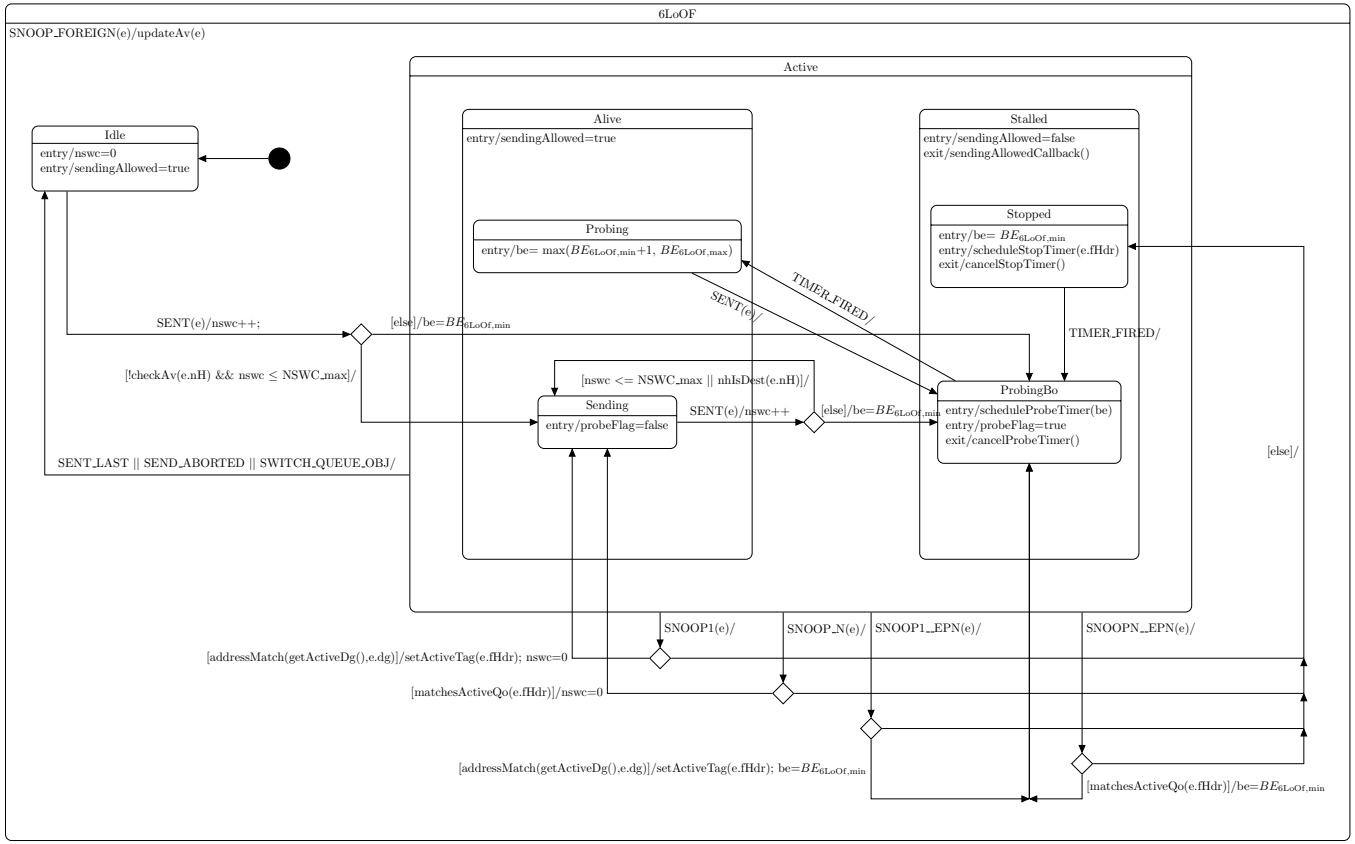


Figure 8.3: UML statechart – Core of the 6LoOF protocol

Short explanations are given for the events used to trigger transitions:

- SNOOP_FOREIGN: a fragment originating from a node different from the current next hop has been overheard
- all other SNOOP-prefixed events denote an overheard fragment originated at the node's current next hop:
  - SNOOP1: a fragment with header FRAG_1, i.e., the first fragment of a datagram
  - SNOOP_N: a fragment with header FRAG_N
  - A trailing _EPN denotes that a modified (see Sect. 8.1.1) fragmentation header has been received
- SENT: a fragment has been sent by the 6LoWPAN layer
- SENT_LAST: the last fragment of a datagram has been sent by the 6LoWPAN layer
- TIMER_FIRED: the stop or probe timer has expired
- SWITCH_QUEUE_OBJ: the 6LoWPAN layer has moved on to transmit a new datagram because the current datagram starved
- SEND_ABORTED: the 6LoWPAN layer has dropped the current datagram, e.g., due to a transmission failure at the link layer
- ENQUEUE_INTERF: a fragment originating at our current next hop was enqueued, potentially deadlocking the queues at both nodes

The used function calls and variables are defined in Table 8.2. In the following, additional mechanisms of 6LoOF and the less obvious entries of Table 8.2 are explained.

Timers

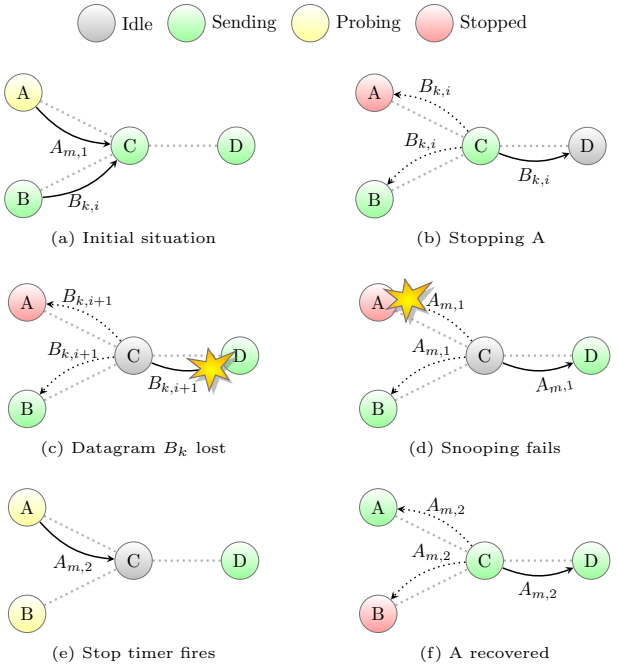
There are two timers involved in the execution of the 6LoOF protocol. These trigger a single event, which is treated differently depending on the active state. The probe timer is used to quickly reduce the frequency of transmissions at a node upon encountering an uncertain situation, i.e., not snooping any relevant fragment. As a basic unit to calculate its delay period, it employs the same EWMA estimate of the average transmission duration as the adaptive rate restriction forwarding mode (see Sect. 5.2.1). Thereby, the probing delay is calculated as

$$T_{\text{probe}} = 2^{be} T_{\text{tx}} k \tag{8.1}$$

with k being a pseudo-random number with $k \in [0.5, 1)$ and T_{tx} the current estimate for the duration of a link layer transmission. The current probe timer exponent be is governed by its initial (minimum) value $BE_{\text{6LoOf,min}}$ and its maximum value $BE_{\text{6LoOf,max}}$.

be	probing exponent; to calculate backoff in probing state
nswc	number of fragments sent without snooping from current next hop
sendingAllowed	output; denotes if 6LoOF allows sending
probeFlag	output; signals probing state to 6LoWPAN module
NSWC_max	6LoOF parameter $n_{\mathrm{NSWC,max}}$
$BE_{\mathrm{6LoOf,max}}$	6LoOF parameter $BE_{\mathrm{6LoOF,max}}$
$BE_{\mathrm{6LoOf,min}}$	6LoOF parameter $BE_{\mathrm{6LoOF,min}}$
e.fHdr	fragment information (size, offset, tag)
e.nH	IEEE 802.15.4address of current next hop
e.dg	IPv6 datagram object
addressMatch()	match IPv6 source and destination address
matchesActiveQo()	match by tag or sanity check (Sect. 8.1.1)
setActiveTag()	attach received 6LoWPAN tag to active datagram
getActiveDg()	returns the currently active IPv6 datagram (cf. Sect. 8.1)
nhIsDest()	returns true, if current next hop equals the destination of the fragment
scheduleStopTimer()	start timer to guarantee leaving state Stopped eventually
scheduleProbeTimer()	schedule probing backoff
cancelStopTimer()	stop the stop timer
cancelProbeTimer()	stop the probe timer
checkAv()	returns true, if a recent transmission from the current next hop has been overheard
updateAv()	set or clear bit for neighbor based on snooped 6LoWPAN frame
sendingAllowedCallback()	signal 6LoWPAN layer that 6LoOF allows sending the next fragment

Table 8.2: Variables and actions (functions) used in the 6LoOF state machine



(a) Initial situation (b) Stopping A

(c) Datagram B_k lost (d) Snooping fails

(e) Stop timer fires (f) A recovered

Figure 8.4: Working of the stop timer; after transmission failure of C (8.4d), A could become suspended infinitely. The stop timer puts A back into ProbingBo (8.4f)

The stop timer is necessary to prevent a node from being stuck in state Stopped. Consider the three nodes in Fig. 8.4. Node A entered state Stopped after snooping fragment $B_{i,k}$. If datagram B_i is lost due to a buffer overflow or consecutively failing transmissions at the link layer, node B will eventually stop sending fragments belonging to it. In consequence, node A has no means to assess the situation without snooping another fragment. One fragment of A_j has to be present at C, because A always transmits at least one fragment in probing state. However, there is no guarantee that A successfully snoops such a fragment when it is transmitted by C. If no fragment is snooped, C starves while A is stuck in state Stopped. Therefore, a node always starts a timer upon entering state Stopped, with a duration of

$$T_{\text{stop}} = \frac{\frac{\text{datagram\_size} - 2^3 \text{datagram\_offset}}{\text{e.ffHdr.size}} T_{\text{tx}} k_{\text{stop}}}{2}, \tag{8.2}$$

$T_{\text{tx}} \times k_{\text{stop}}$ represents the average waiting time between forwarding two fragments with the Direct-ARR mode (Sect. 5.2.1). T_{stop} is half the estimated time it would take this node to finish the transmission of the current next hop's active datagram. If this timer is triggered, the nodes enters state ProbingBo to transmit new fragments and eventually be able to snoop a confirming transmission.

Initial Transitions

Before some datagram arrives at a node, the node remains in state Idle. Another important decision concerns the state to enter upon sending the first datagram. To simplify this decision, nodes permanently monitor their neighborhood for transmissions. Ideally, a node stores information on individual transmissions of all neighbors, including the information necessary to calculate a stop timer delay as given in (8.2). It could then enter state Stopped directly without causing any interference to the ongoing transmission of its next hop. First, however, this would incur additional memory costs (for at least datagram_size and datagram_offset), growing linearly with the size of the routing table, i.e., the number of potential next hops. Secondly, directly entering state Stopped denies a node the chance to leave this state by snooping one of its own fragments (because it never sent any) in case its current next hop drops the datagram that was responsible for letting the node enter state Stopped.

Therefore, 6LoOF only flags each entry in its routing table with an activity bit. updateAv() sets this bit if any FRAG_N header is snooped that does not denote the last fragment of a datagram, in which case the bit is cleared. Receiving an event SENT with a node that is flagged active as next-hop destination, a node enters ProbingBo instead. Note that this single-bit flag has a good chance to not occupy any additional memory, because an entry in a routing table usually contains a variable for prefix length, which can be restricted to 7 bit (sensible prefix lengths are in the range $[1, 128]$).

If the current next hop is considered idle, the 6LoOF parameter "**n**umber **s**ent **w**ithout **c**onfirmation" (NSWC_max) governs how long a node will stay in state Sending if it does not snoop its own fragments being forwarded by the current next hop. NSWC_max specifies the number of fragments to transmit without such a confirmation (cf. Fig. 8.3).

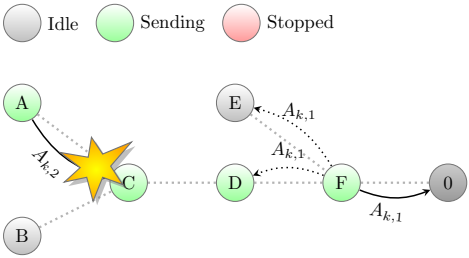


Figure 8.5: Scenario leading to a node starving; new incoming datagrams are blocked at node F, because no new fragments arrive for datagram A_k, which is at the head of the transmission queue

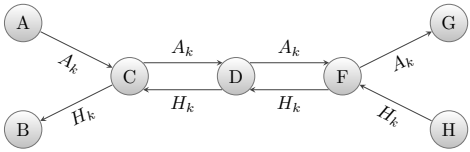
Prevention of Node Starving

Because 6LoOF forwards datagrams in an orderly fashion, a mechanism has been implemented to prevent nodes from "starving unnecessarily". Different from the straightforward direct modes, a node always tries to forward all fragments belonging to one datagram before moving to another (Sect. 8.1). This property makes the idea of suspending transmissions viable in the first place. A fragment of datagram A_k that is lost at a former node on the route, however, causes a node farther down the route to block on that datagram. This is illustrated in Fig. 8.5, where node F is starving after transmission of $A_{k,1}$, because $A_{k,2}$ has been lost at C. The active datagram A_k is at the head of the transmission queue at node F and no new fragments are going to arrive. Note that node F has no way of knowing that the datagram A_k has been lost. If node E starts transmission of a datagram, this new datagram will be enqueued after A_k and is blocked until either datagram A_k is removed due to a timeout or is removed from the head of the queue by some other means.

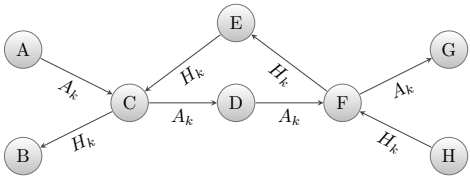
Waiting for the regular timeout in the order of seconds to take care of the starving datagram is clearly undesirable. The situation is therefore resolved by allowing a node to postpone the transmission of such a datagram after it encountered an empty active datagram for a specified number of times, governed by the 6LoOF parameter queueSwitchAfter (n_{qsa}). Thereby, datagrams that are already lost are put to the end of the queue and eventually timeout without blocking active datagrams. If, on the other hand, the datagram is still alive, i.e., the lack of fragments has been caused by mere delays and not losses, the transmission can still be completed at a later time.

In general, a node always tries to transmit the next fragment immediately (Direct), after a delay (Direct-ARR) or when being allowed to transmit by the 6LoOF module. If this first attempt fails, because the active datagram is not able to provide the next fragment, 6LoOF recoils and waits for the next fragment to arrive. This method achieves that datagrams are only pushed back to the queue if there are other datagrams waiting to be sent. Senders of these other datagrams, however, enter state ProbingBo on not snooping any of their fragments. Thus the rate of incoming fragments is governed by the probing algorithm.

To reduce the reaction time in such situations, I also experimented with a method that instead schedules a new transmission attempt after a period of T_{tx} has elapsed.



(a) Example for special case with trivial cycle



(b) Example for general case with arbitrary-length cycles

Figure 8.6: Emergence of a Deadlock in 6LoOF with routes for two different datagrams originating at nodes A and H

Preliminary tests, however, have shown that this variant does not achieve an improvement in terms of reliability or latency and is therefore not considered further.

In addition to the presented push back scheme, 6LoOF also checks incoming fragments for being a successor of the currently active datagram. This is the case if link layer source and destination addresses match and the incoming inbound tag is larger than the inbound tag of the active datagram. Receiving a datagram with that younger tag implies that the node's current predecessor also has switched to forwarding a different datagram. If in this situation the active datagram does not have any fragments available for transmission, it is pushed back to the end of the queue, because no new matching fragments will be received for this datagram.

Note that a similar issue may also appear the other way around, i.e., consider node F dropping A_k, e.g., due to an unsuccessful transmission towards node 0. In this case, node D can never snoop a confirming fragment and will transmit at snooping rate, it is snoop-starving. A fragment being lost in spite of link-layer retransmissions in many cases indicate the occurrence of a large traffic volume leading to a crowded wireless channel or buffer drops. Reducing the transmission rate in those cases is reasonable. However, in some cases this behavior is potentially too conservative.

A possibility to allow a predecessor node to be informed about such a lost datagram is the transmission of a dummy fragment upon reception of a fragment that belongs of the dropped datagram. However, a modification to the existing header types would have to be used in that case, similar to the proposal in Table 8.1. Unfortunately, this issue and the potential mitigation were discovered too late in the process of completing the dissertation to be incorporated into the evaluation.

Deadlocks

Another important consideration is deadlock prevention. As shown in Fig. 8.6, it is enough for two nodes to route datagrams along the same path in opposite directions to create a deadlock. Both nodes hear the other node forwarding some datagram different from their own and enter state Stopped. Only after a timeout they transmit another fragment, eventually snoop the other node's fragment and enter state Stopped again. Note that the two problematic nodes do not have to directly send to each other for the problem to occur (Fig. 8.6b).

It is assumed that each route in a network forms a directed acyclic graph. If the graph resulting in the union of all routing graphs contains a cycle, then there are at least two nodes in danger of a deadlock. In Fig. 8.6b, node D may stop on overhearing the transmission $H_{k,i}$ from node F and node E will stop on overhearing $A_{k,j}$ from node C. The major issue with the general case is, that it cannot be detected by either node without extensive additional information about the history of a datagram.

For this work, however, I assume that RPL ([Win+12]) is the routing protocol of choice for typical applications in LLNs. Considering the scope of 6LoWPAN and the focus on collection traffic in this thesis, this is a reasonable assumption (see Sect. 2.2.4). RPL always creates a routing tree and routes upstream and downstream traffic along the same branch of that tree. Even one-to-one traffic is routed along the tree by routing up towards a common ancestor and from there down to the destination. While the tree may become corrupted at some point and temporary routing loops may form, the protocol employs means to recover from the failure. In such a situation, a deadlock in 6LoOF is not the most urgent problem to solve. Therefore, it is assumed that the routing protocol above 6LoOF forms routing paths that result in a network topology (by union of all paths) that contains only trivial circles.

On this assumption, 6LoOF employs a simple deadlock recognition and avoidance strategy. For every incoming fragment, a node checks if this fragment originates at the next-hop destination of its active datagram. A reverse lookup in the neighbor discovery's neighbor cache ([Nar+07]) can be employed for this task. If the incoming fragment indeed originates from the current next hop, the IPv6 source address of incoming and active datagram is compared to decide which datagram to forward. In that case, the datagram originating at the node with the smaller address proceeds, while the other datagram is rotated to the end of the transmission queue.

Edges of the Network

A node performing the final transmission of a datagram towards the IPv6 destination or the edge of the wireless network can not rely on snooping to govern its transmission behavior. Thus, nodes determining that next hop and destination address are equal always stay in state Sending for the duration of the transmission. In general, the destination may also be located outside the 6LoWPAN wireless network. Assuming the usage of the RPL protocol as above, the DODAG root can be used to catch those cases and to reliably determine whether to enter the "send always" mode.

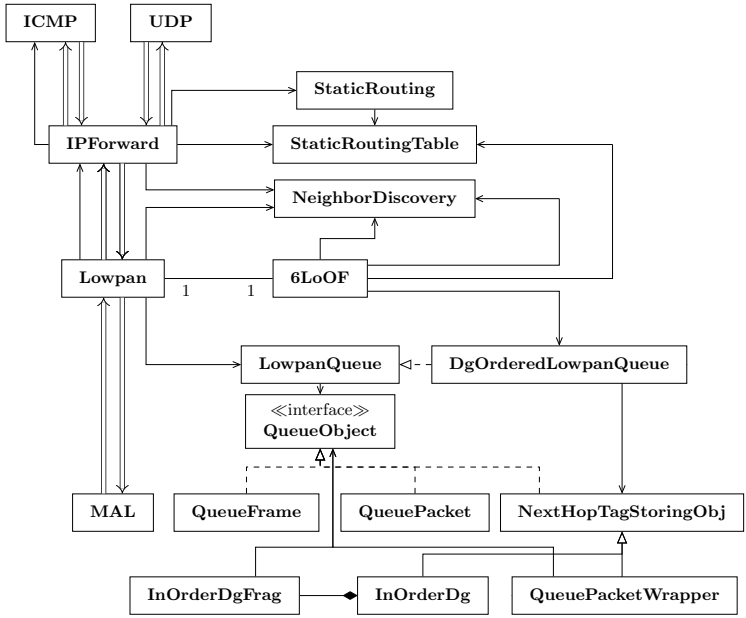


Figure 8.7: 6LoOF implementation in CometOS

8.2 Implementation

I implemented 6LoOF for CometOS [UWT12][1]. The 6LoOF module is integrated (Fig. 8.7) into the refactored 6LoWPAN IPv6 stack (Sect. 5.3).

The `LowpanOrderedForwarding` module is responsible to execute the actual state machine. The `Lowpan` module generates events, transmits the actual fragments and manages the queue functions (Sect. 8.1.3). Before transmitting, it always checks if it is allowed to do so by the `LowpanOrderedForwarding` module. 6LoOF-specific functionality is localized in sending and enqueueing methods of the `LocalCongestionAvoider` module, the `Lowpan` module and the `LowpanQueue` (Fig. 8.7). The latter two can be replaced by objects of different subclasses and the former part is only active if these are realized by the corresponding 6LoOF objects. This design allows 6LoOF to be (de-)activated at runtime.

The 6LoOF implementation alters the behavior of the 6LoWPAN queue by providing a new subclass for `LowpanQueue` and new subtypes of `QueueObjects` (Fig. 8.7). The `DgOrderedQueue` stores `NextHopStoringQueueObject` objects instead of `QueueObjects`. `NextHopStoringQueueObject` add the ability to attach 6LoWPAN tags to datagrams as they are snooped by 6LoOF. Datagrams originating at a node are therefore also wrapped by `QueueObjectWrapper` objects. While the plain 6LoWPAN implementation

[1] https://github.com/CometOS/CometOS

object type	additional RAM used
`DgOrderedQueue`	7 B (more flexible queue structure)
`QueueObjectWrapper`	5 B per object
`InOrderDg`	9 B per object
`InOrderDgFrag`	4 B per object

Table 8.3: 6LoOF-specific objects

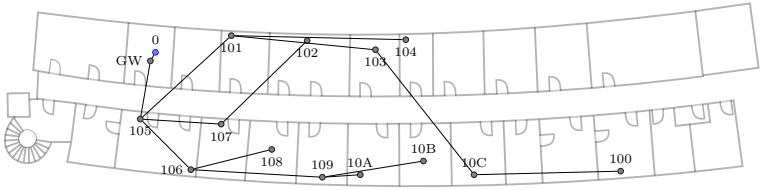


Figure 8.8: TB-D, location of nodes and routing tree for TB-D and RS-D1

enqueued incoming fragments as `QueueFrame`s individually, for 6LoOF they are matched by their tag and stored as a linked list of `InOrderDgFrag` in `InOrderDg` objects. Thereby, the ordering of datagrams is realized.

Compared to the plain `Lowpan` implementation, 6LoOF requires additional program and data memory, which can be attributed mainly to the necessity to keep track of fragmentation header tags used by next hop nodes and the described wrapper objects. As individual implementations may vary, the data for the CometOS implementation is provided in Table 8.3.

8.3 Experiment setup

The setup for the employed testbeds as well as the configuration of all simulation scenarios for the evaluation of the 6LoOF protocol is described in this section.

8.3.1 Testbeds

Two different testbeds were employed for the evaluation of 6LoOF. In addition to a variant of the testbed at the Hamburg University of Technology (cf. Sect. 7.1.1), another one located at the IoTLab in France [Fam+14; Adj+15] was used.

At Hamburg University of Technology

First, I facilitated another variant of the Telematics testbed, similar to the ones used for evaluation of the basic and enhanced forwarding modes (cf. Sect. 5.4.1 and 7.1.1). Different from these formerly used testbeds, it employs a serial-to-wireless bridge node attached to the basestation (i.e., the script controlling the experiment, cf. Sect. 5.4.1) via the SerialComm instead of a node that contains a complete network stack. The setup was slightly changed in that way because the simulation model was not able to

adequately capture the transparent serial link on the last hop (cf. Sect. 7.2.2). With the new setup, there still exists the additional serial connection, but the setup above layer two is equal. Furthermore, hacks to circumvent the problem that the last hop communication could not be snooped become unnecessary. The location of nodes in the corridor of our office building and the resulting routing tree are shown in Fig. 8.8.

The serial-to-wireless bridge complicates time synchronization between basestation and neighboring nodes because any `InitialMessage` or `TimestampMessage` may be buffered in the forwarding queue. To retain the capacity for time synchronization over the bridged link, the TimeSyncWirelessBridge module inspects incoming frames, determines their service time and modifies the frame correspondingly. Furthermore, to configure the parameters of the IEEE 802.15.4 CSMA/CA of the bridge node's transceiver, a dedicated remote access module is installed. The distinction between frames to forward and frames destined to the bridge node itself is made by their protocol dispatch identifier paths.

At IoTLab

Secondly, I employed the IoTLab, specifically the site at Grenoble, to conduct larger-scale experiments with the 6LoOF protocol. CometOS, including the Software MAC, was ported to the M3 OpenNode platform used in the IoTLab. This wireless module is comprised of a STM32F103REY ARM Cortex M3 MCU and the Atmel AT86RF231 wireless transceiver. The M3 OpenNode nodes assigned to an experiment expose their UART interface via a TCP connection to the user. To prevent the issue with inconsistent transmission power in the Telematics testbeds caused by the use of the wireless link for control traffic and time synchronization with maximal transmission power (cf. Sect. 5.4.1), the connections of the protocol stack were slightly changed to use those serial connections instead of wireless links. Furthermore, the AODV implementation used for the control stack's routing layer did not scale very well to the larger network and encountered issues with regard to its reliability and performance.

Tunneling the packetized SerialComm including its simple stop-wait automatic repeat request (ARQ) via TCP from a local machine via the SSH gateway of IoTLab site at Grenoble to the M3 OpenNode nodes proved to be prohibitively slow, especially the connection towards the M3 OpenNode employed as bridge node. Another alternative would have been to execute the basestation script at the SSH gateway of the Grenoble site, but cascading incompatibilities (starting with the available version of Python) ruled out this opportunity.

Apart from the M3 OpenNode nodes, the IoTLab also has some more powerful A8 Nodes installed, that run an embedded Linux, which proved powerful enough to run the basestation script. The A8 Nodes are also serially connected to a slightly modified M3 OpenNode located on the same board. This M3 OpenNode node was used as wireless bridge. Because M3 OpenNodes and A8 Nodes are located in different sub-networks and are not allowed to communicate with each other directly, a complex forwarding scheme had to be installed to establish a link from the basestation to the M3 OpenNodes serial interface for experiment control.

The IoTLab experiment was created and initialized via the IoTLab-CLI from a local machine at Hamburg University of Technology. In addition to acquiring and flashing the necessary set of nodes at the Grenoble site, this initialization script triggers several actions at the SSH gateway of the Grenoble Site:

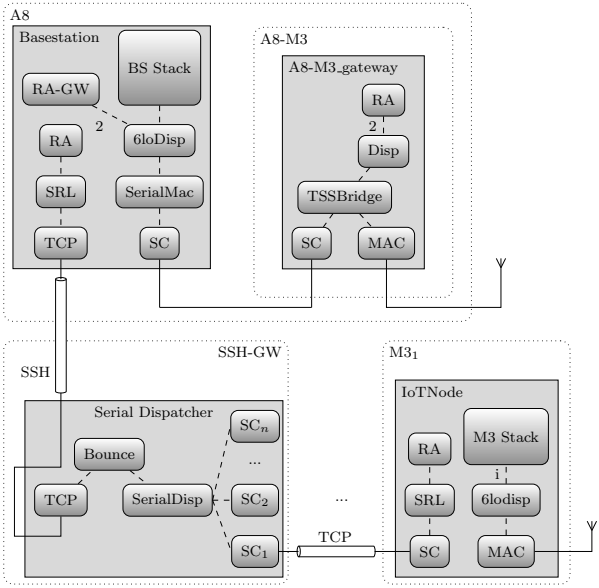


Figure 8.9: Experiment setup for the IoTLab

- Create virtual serial interfaces via socat, which connect via TCP to the available TCP port of each M3 OpenNode, tunneling the serial connection via TCP.

- Launch the application `SerialDispatch`, which dispatches all frames from the basestation to the serial connection corresponding to the destination address and vice versa. The mapping between serial interfaces and addresses (IoTLab uid) is realized by the IoTLab node id.

- Start collection of wireless sniffer data (cf. Sect. 8.3.1).

- Wait for the A8 Node node to boot and launch the actual basestation script.

The basestation in turn establishes a connection to the `SerialDispatch` application, wires its control stack to the corresponding TCP module and starts execution of the configured experiment runs. The setup resulting from those actions is shown in Fig. 8.9.

The wireless links at the IoTLab proved to be much more stable, yielding less fluctuating network topologies. Hence, I was able to perform the actual experiment and the collection of topology information with the same transmission power. The methodology used was the one described in Sect. 4.2.3, with $\mathrm{RSSI}_{\mathrm{th}} = -80\,\mathrm{dBm}$, $k_{\mathrm{abs}} = 0.1$ $k_{\sigma^2} = 0.002$.

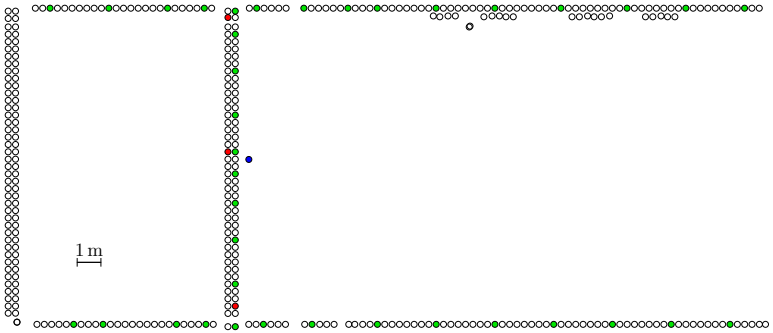


Figure 8.10: Nodes selection at Grenoble site comprising the TB-IoT network; green: experiment nodes, blue: basestation and gateway node, red: sniffer nodes

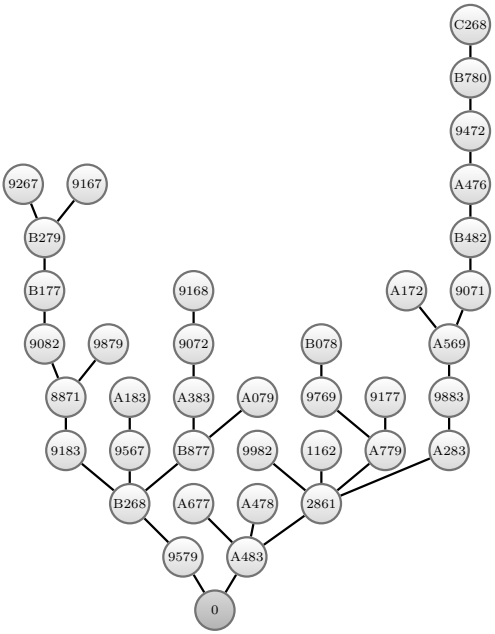


Figure 8.11: Static routing topology of TB-IoT and RS-IoT networks

			none	foreign	own
	RSSI	$= -90\,\text{dBm}$	n_{idle}	$n_{\text{f,idle}}$	$n_{\text{o,idle}}$
$-90\,\text{dBm} <$	RSSI	$\leq -80\,\text{dBm}$	n_{90}	$n_{\text{f},90}$	$n_{\text{o},90}$
$-80\,\text{dBm} <$	RSSI		n_{80}	$n_{\text{f},80}$	$n_{\text{o},80}$

Table 8.4: Counting categories for sniffer nodes resulting from the two dimensions signal strength (columns) and decoded IEEE 802.15.4 signal (rows). For the latter, 'none' indicates no IEEE 802.15.4 signal, 'foreign' and 'own' a decoded signal with a different and the same IEEE 802.15.4 network ID, respectively

Radio Sniffers

The 2.4 GHz ISM band at both locations was not or only partially under my control. Both locations are potentially exposed to a large amount of IEEE 802.11 traffic. At TB-D, I could rule out other IEEE 802.15.4 traffic using the same channel, at the IoTLab testbed, this was not possible. Therefore, I additionally installed several sniffer nodes at both location that checked the channel for ongoing IEEE 802.15.4 transmissions and other traffic. The sniffers periodically measured the current RSSI on the channel.

Each measurement is categorized according to the two dimensions signal strength and type of signal as shown in Table 8.4. Corresponding counters keep track of the number of measurements in each category. To support categorization, the sniffer nodes use a special stripped-down radio driver. The type of signal is distinguished in two stages. First, every measurement that is not taken between an RX_START and RX_-END interrupt is put into the 'none' category. Then, the destination PAN id field of the IEEE 802.15.4 MAC header's frame control field of the IEEE 802.15.4 MAC header (FCF) is used to categorize if the frame belongs to my experiment or to an experiment of another user, which uses the same channel. The signal strength dimension is directly derived from the RSSI value as reported by the Atmel ATmega256RFR2 or Atmel AT86RF231.

Because the RX_START interrupt and the MAC header cannot be decoded instantly to determine the type of signal, some measurements cannot be categorized immediately. These are instead buffered until the decision can be made. Furthermore, acknowledgment frames do not carry a MAC header and are instead categorized according to the last snooped data frame. The mechanism is not perfect. Acknowledgments may be matched wrongly and more errors can be introduced by collisions of frames. In case of a collision of frames, only one or even no SFD can be decoded, causing the measurements to be erroneously categorized as 'none'. However, the purpose of the sniffers is to notice significant traffic from other networks in the area. Test with a co-located IEEE 802.15.4 network using a different PAN id have shown that number of measurements categorized as foreign for such a setup is several orders of magnitude higher than the numbers produced by wrong categorization.

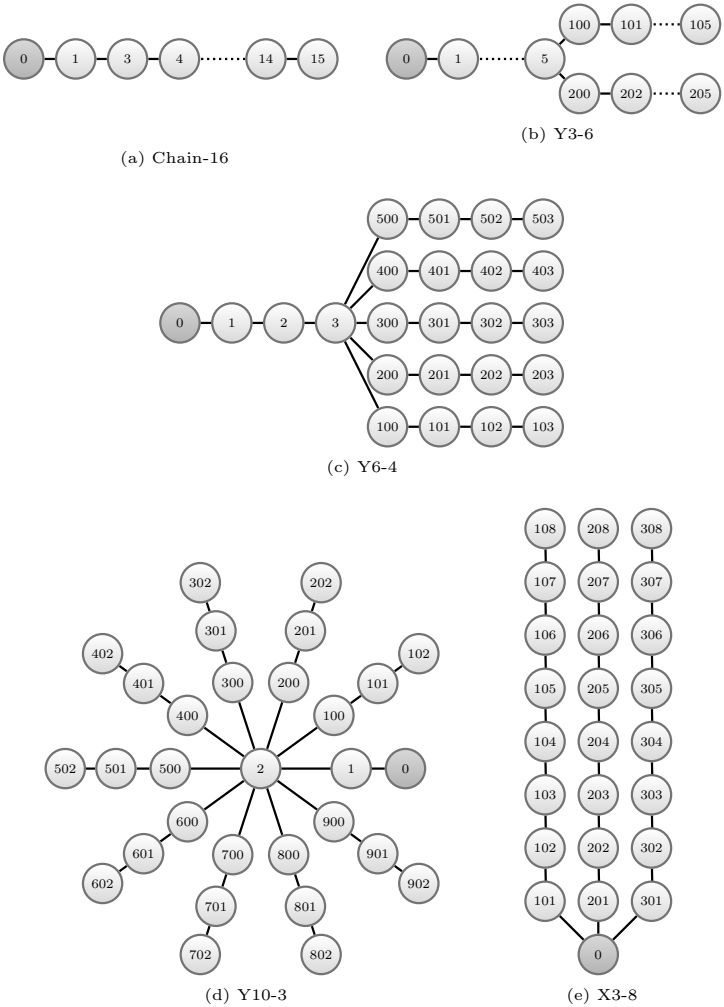
	Assembly	Direct	Direct-LOOF
data buffer	3840	3840	3584
`AssemblyHandler`	585	121	121
`QueuePacket`	390	78	78
`FifoLowpanQueue`	19	19	0
queue size	20	120	72
buffer handlers	300	300	280
`DirectHandler`	0	184	183
`QueueFrame`	0	540	504
`DgOrderedQueue`	0	0	26
`QueueObjectWrapper`	0	0	10
`InOrderDg`	0	0	135
`InOrderDgFrag`	0	0	224
6LoWPAN to IPv6messages	580	116	116
IPv6 to 6LoWPAN messages	480	900	900
total	6214	6227	6243

Table 8.5: Memory usage of CometOS 6LoOF implementation in B for the different forwarding modes as used during experimental/simulative evaluation of the 6LoOF protocol

8.3.2 Memory Usage

As explained in Sect. 8.2, 6LoOF occupies some additional memory compared to the other forwarding modes. To allow for a fair comparison, the size of the data buffer was adjusted to make up for the additional memory required by the 6LoOF implementation. The total memory demand may vary among different implementations of 6LoWPAN and 6LoOF.

The memory usage of the CometOS IPv6 stack with and without 6LoOF is shown in Table 8.5. The Assembly mode has to reassemble each datagram at each hop, therefore, it needs a larger number of metadata structures for that purpose, which is reflected in the larger memory demand for the `AssemblyHandler`. The Direct forwarding modes, on the other hand, potentially need a larger 6LoWPAN queue and a `QueueFrame` object for every incoming fragment (`QueueFrame` and queue size). The CometOS implementation uses dynamic memory allocation for `QueueFrame`s and message type objects, in which case the maximum memory demand is given. 6LoOF, on the other hand, needs additional wrapper structures and metadata (cf. Sect. 8.2), which occupy some additional memory. Therefore, the size of the data buffer for 6LoOF was chosen to be 256 B smaller than that of the other forwarding modes. Thereby, the memory demand of all forwarding modes is about equal. Note that this does also apply for other sizes of the data buffer, because the other implementation parameters are independent of the size of the data buffer.



(a) Chain-16

(b) Y3-6

(c) Y6-4

(d) Y10-3

(e) X3-8

Figure 8.12: Network routing trees for the idealized network topologies. Edges represent static routes, the dark gray node is the sink.

8.3.3 Simulation Environment

The idealized network topologies employed in the evaluation of the 6LoOF protocols are shown in Fig. 8.12. Those are similar to the networks presented in Chapter 5, but care was taken that the overall number of hops when summing up the length of paths of all nodes to the sink is about equal. In addition to the "Y-networks" (Y3-6, Y6-4 and Y10-3), which exhibit a single bottleneck node and a different number of arms of different lengths, a long chain of nodes and a network with three arms that only interfere at the sink were used (Chain-16 and X3-8). As before, the links in all of these networks were idealized, i.e., the shown routing topology corresponds to the transmission and interference range. Each wireless link is set to an average RSSI of −88 dBm with a variance of 49 dB.

8.4 Evaluation: 6LoOF vs Plain Forwarding

This section presents the results of the experimental and simulative evaluation of the 6LoOF protocol. First, the influence of the 6LoOF parameters is investigated with a simulative parameter study in the RS-C network. Secondly, results of experiments in testbeds at Hamburg University of Technology and the IoTLab are shown. Those results are used to validate corresponding simulations, which additionally investigate a wider parameter space and additional, idealized network topologies.

The IEEE 802.15.4 MAC and 6LoOF parameters were fixed to values determined in the corresponding parameter studies (Sect. 7.3 in Table 7.5.

The traffic pattern used for this evaluation is the one described in Sect. 5.4.4, i.e., a periodic data collection with transmission intervals that were randomized by a uniform distribution.

Results that represent a counter, e.g., the number of dropped fragments or datagrams or the number of fragments that underwent a certain number of retransmission attempts, in some cases are normalized. Unless stated otherwise, the number of requested UDP transmissions for that experiment is used as normalization factor. Thereby the counter values become comparable between runs with different UDP packet size (L_{UDP}) and/or a different total number of packets. Some counted values are depicted as stacked bar plots. With these, each segment represents the share of dropped fragments a certain failure is responsible for. The segments are not ordered by their size and the topmost segment does **not** implicitly stretch over the other segments below it.

8.4.1 6LoOF Parameters

The influence of the 6LoOF parameters $n_{\mathrm{NSWC,max}}$, $BE_{\mathrm{6LoOF,max}}$, $BE_{\mathrm{6LoOF,min}}$, n_{qsa} and x_{EPN} as well as that of the 6LoWPAN timeout T_{to} was evaluated in the simulation environment using the RS-C network topology (see Sect. 7.1.2), i.e., variant which used equal transmission power for the experiment and the construction of the topology.

The set of parameters is shown in Table 8.6. Note that the configuration of the IEEE 802.15.4 MAC layer does not change for the remaining 6LoOF evaluation presented in this section and is chosen corresponding to the results of the parameter study presented in Sect. 7.3.

forwarding mode Dir-LOOF, Dir-ARR-LOOF	$BE_{\text{6LoOF,min}}$ 1,2,3	$BE_{\text{6LoOF,max}}$ 4,5,6,8	$n_{\text{NSWC,max}}$ 1,2,4,8	Q [B] 3584
macMaxFrameRetries 7	macMinBe 5	macMaxBe 8	macCcaMode 0	L_{UDP} 1200
queueSwitchAfter 1,2,3,8	useEpnFlag 0 (no), 1 (yes)	λ_{B} [B s^{-1}] 112.5,150	T_{to} [s] 2,3,4,5,6	

Table 8.6: Parameter values for 6LoOF parameter study

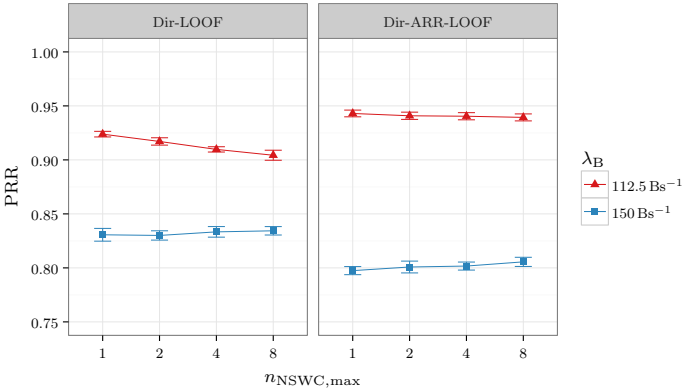


Figure 8.13: PRR against $n_{\text{NSWC,max}}$ parameter for different values of λ_{B} and forwarding modes (columns), and $BE_{\text{6LoOF,min}} = 2$, $BE_{\text{6LoOF,max}} = 5$, $n_{\text{qsa}} = 3$; $n_{\text{NSWC,max}}$ does have only small influence on the overall PRR

Due to the large parameter space, it is not possible to provide an all-encompassing overview about the results of all combinations of parameters. Therefore, only the significant results are provided, fixing other parameters to retain clarity.

NSWC_max: $n_{\text{NSWC,max}}$

Figure 8.13 shows the influence of the parameter $n_{\text{NSWC,max}}$, which governs the number of fragments to transmit without receiving a snooped confirmation before a node enters the state ProbingBo. We can see that the overall influence of the parameter on the average PRR over all nodes is very limited. The most pronounced change in PRR can be observed for the combination of using the Direct-LOOF forwarding with the lower application payload data rate of $\lambda_{\text{B}} = 112.5\,\text{B}\,\text{s}^{-1}$. For this configuration, the average PRR drops from 0.924 to 0.904 for values of $n_{\text{NSWC,max}}$ of 1 and 8, respectively. However, for the higher application data rate of $\lambda_{\text{B}} = 150\,\text{B}\,\text{s}^{-1}$, an nearly inverse trend can be observed. This observed limited influence can be explained by two facts:

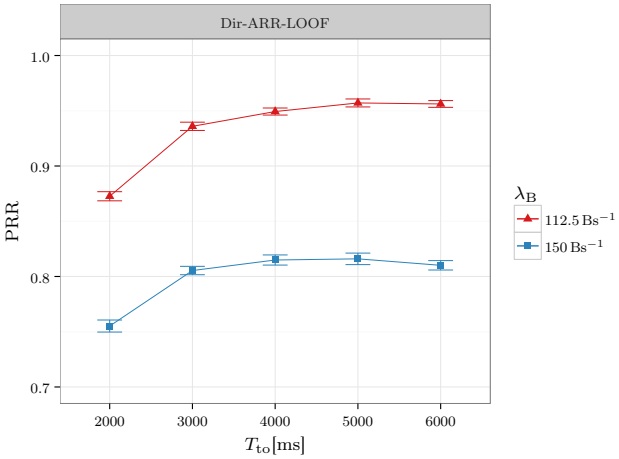


Figure 8.14: PRR of Direct-ARR-LOOF mode against T_{to}; $BE_{6LoOF,min} = 2$, $BE_{6LoOF,max} = 5$, $n_{NSWC,max} = 1$, $n_{qsa} = 3$

- Nodes continuously monitor their vicinity and check for activity of their neighbors before transmitting and enter state Probing upon beginning a new transmission and finding their current next hop busy.

- The parameter only influences the behavior if no fragment is snooped, i.e., the current next hop is in state Stopped or snooping fails.

Considering these results $n_{NSWC,max}$ was fixed to 1 for the remaining evaluation.

6LoWPAN Timeout: T_{to}

As shown in Fig. 8.14, the 6LoWPAN timeout T_{to} (cf. Sect 5.3) has a strong influence on the average PRR for the evaluated setup. For $T_{to} = 2000$ ms, as used with the plain and enhanced Assembly and Direct forwarding modes, the Direct-ARR-LOOF mode exhibits a PRR that is about 0.08 lower than the PRR for $T_{to} = 5000$ ms. This indicates that a non-negligible number of datagrams remains at some node for longer than the timeout period.

A possible explanation of this observation can be found in the inverse of the "node starving"-problem described in Sect. 8.1.3: A fragment of a datagram is lost on a subsequent hop, a node situated ahead of the node that lost the fragment will stop to receive confirmations by snooping. Furthermore, this situation is indistinguishable from a node that has merely entered state Stopped and only waits for a following link to become available again and from failed snooping of fragments. Therefore, the node waiting for an opportunity to forward it's fragment can only do so at the snooping rate, when no other snooped fragments cause it to enter state Stopped itself. This

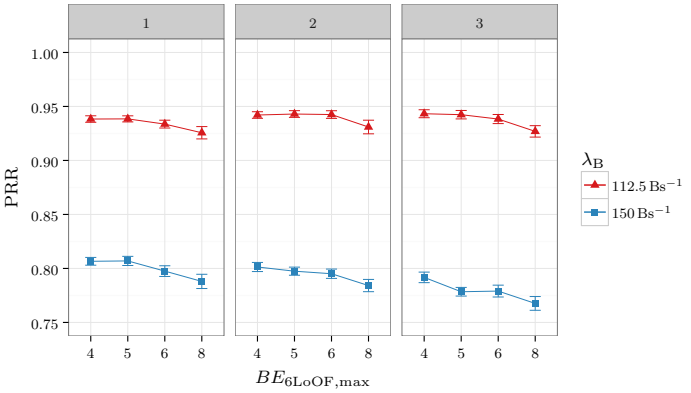


Figure 8.15: PRR of Direct-ARR-LOOF mode against $BE_{6\text{LoOF,max}}$ for different value of λ_B and $BE_{6\text{LoOF,min}}$ (columns); $n_{qsa} = 3$, $n_{NSWC,max} = 1$.

issue is also suspected to be one of the major problems of the 6LoOF mechanism in its form as presented in this thesis.

Minimum and Maximum Probing Exponents: $BE_{6\text{LoOF,min}}$, $BE_{6\text{LoOF,max}}$

Examination of the backoff exponents of the 6LoOF probing timer also revealed a limited influence for the given network. The results for Direct-ARR-LOOF with a buffer of $Q = 3584\,\text{B}$ and $n_{qsa} = 3$ are shown in Figure 8.15, which governs when the 6LoOF protocol pushes a datagram to the end of its transmission queue after this datagram was not able to provide new fragments (cf. Sect. 8.1.3). The PRR is lower for the larger examined values of $BE_{6\text{LoOF,max}}$, independently from λ_B and $BE_{6\text{LoOF,min}}$. The overall difference between $BE_{6\text{LoOF,max}} = 4$ and $BE_{6\text{LoOF,max}} = 8$ is less than 0.025. The overall influence of the $BE_{6\text{LoOF,min}}$ parameter on the PRR even less pronounced. For the higher of the two data rates $\lambda_B = 150\,\text{B}\,\text{s}^{-1}$, the PRR curve is shifted to lower values for $BE_{6\text{LoOF,min}} = 3$. For the lower data rate of $\lambda_B = 112.5\,\text{B}\,\text{s}^{-1}$, $BE_{6\text{LoOF,min}} = 2$ exhibits the highest PRR. Similar to the $n_{NSWC,max}$ parameter, the influence on the PRR is limited. $BE_{6\text{LoOF,min}} = 2$ and $BE_{6\text{LoOF,max}} = 4$ were chosen as parameters for further evaluation.

Push Back Stale Datagrams: n_{qsa}

Similar observations can be made for the n_{qsa} parameter (Fig. 8.16). For the higher of the examined data rates $\lambda_B = 150\,\text{B}\,\text{s}^{-1}$, the highest PRR is achieved for $n_{qsa} = 1$, the lowest for $n_{qsa} = 8$. With $\lambda_B = 112.5\,\text{B}\,\text{s}^{-1}$, in contrast, the lowest PRR is observed for $n_{qsa} = 1$. $n_{qsa} = 3$ was chosen as a compromise.

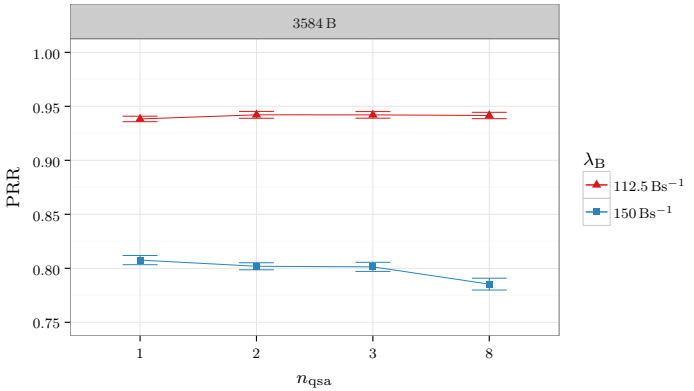


Figure 8.16: PRR of Direct-ARR-LOOF mode against n_{qsa} for two different values of λ_{B}; $BE_{\mathrm{6LoOF,min}} = 2$, $BE_{\mathrm{6LoOF,max}} = 4$

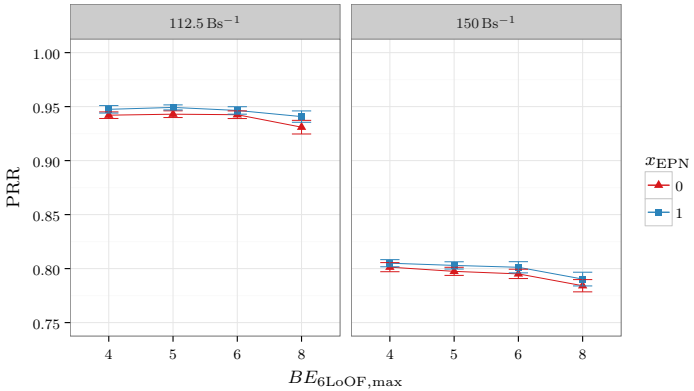


Figure 8.17: PRR of Direct-ARR-LOOF mode against $BE_{\mathrm{6LoOF,max}}$ with x_{EPN} set and cleared for varying λ_{B}; $BE_{\mathrm{6LoOF,min}} = 2$

$BE_{\text{6LoOF,min}}$	$BE_{\text{6LoOF,max}}$	n_{qsa}	$n_{\text{NSWC,max}}$	T_{to} (ms)
2	4	3	1	5000

Table 8.7: 6LoOF parameters used for evaluation

Q [B]	L_{UDP} [B]
3840 (3584)	1200

Table 8.8: Traffic generator parameters for 6LoOF evaluation in testbed

useEpnFlag: x_{EPN}

Figure 8.17 shows the impact of explicit probing notification. As can be seen, it is rather limited. Only for $BE_{\text{6LoOF,max}} = 8$ the PRR without using the flagging mechanism (cf. Sect. 8.1.2) is more than 0.01 lower than the corresponding PRR using the mechanism. For the other observations the PRR is lower by less than 0.01. These findings support the hypothesis that flagging outgoing fragments does not greatly improve the performance of 6LoOF in terms of reliability. In the light of those results, not using the flagging mechanism is clearly preferred, especially as it allows an implementation of 6LoOF without alteration to the 6LoWPAN fragment header format.

6LoOF Parameters: Summary

This section examined the influence of the parameters defined for the 6LoOF protocol on the average PRR for the examined traffic scenario and forwarding modes. One reason for the limited impact is that they change the behavior of 6LoOF mainly in situations that represent an error case. For example, the n_{qsa} parameter influences the handling of the 6LoWPAN transmission queue only if subsequent fragments belonging to the active datagram are missing. Apart from nodes that encounter a stopped node as their next hop, probing is used only when snooping fails.

The 6LoOF parameters are fixed for the comparative evaluation of 6LoOF with the other forwarding modes to the values shown in Table 8.7.

8.4.2 TB-IoT Experiments

Due to the contention for the nodes in the IoTLab with other researchers and a comparatively large number of runs to get sound results for both testbed experiments, the configuration of the traffic generator was restricted to the values shown in Table 8.8. x_{EPN} was varied for both testbeds. The transmission rate of UDP packets λ_{B} was constant within each testbed but adapted to the examined network topology.

During the experiment at the Grenoble site of the IoTLab, nodes used the same transmission power that was used to determine the network topology (cf. Sect. 8.3.1). λ_{B} was set in a way to produce a total network traffic (considering all paths of all nodes as in (7.2)) of $\lambda_{\text{B,total}} = 6600\,\text{B}$. This configuration is shown in Tbl. 8.9.

λ_B[B s^{-1}]	P_{tx} [dBm]	x_{EPN}
37.5	-2.5	0,1

Table 8.9: Parameter setting for TB-IoT experiment

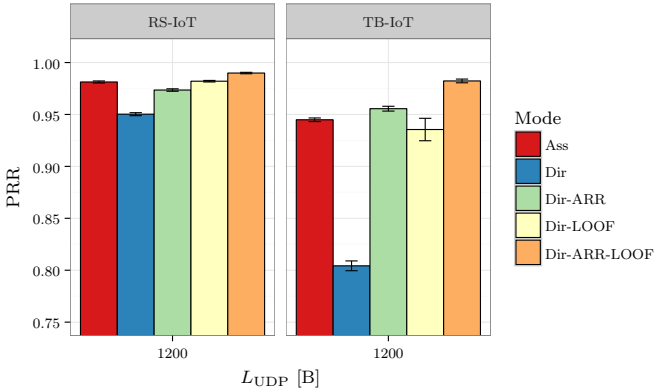


Figure 8.18: PRR of TB-IoT testbed and derived simulation scenario; $x_{EPN} = 1$

PRR in Comparison to Simulation

Figure 8.18 shows the PRR obtained in the IoTLab and a corresponding simulation scenario derived using the method described in Sect. 4.3. The results show that for the given configuration, the 6LoOF mechanism achieves the best performance of all forwarding modes in terms of PRR, in simulation and testbed. With a PRR of 0.982 the PRR obtained by the Direct-ARR-LOOF mode is larger by 0.037 than that of the Assembly mode (0.945) and by 0.026 than that of the Direct-ARR (0.956). To the plain Direct mode, the gap is significantly larger. Note that while the difference seems small it can be also translated to losing less than half as many datagrams than the next-best forwarding mode. The Direct-LOOF mode, with exception of the plain Direct mode, achieves a lower PRR than the other modes.

Comparing the results from the testbed with the corresponding simulation, the most notable difference is the better PRR exhibited by Direct and Direct-LOOF modes. The Direct-LOOF forwarding achieves a PRR slightly higher than that of the Assembly mode and also outperforms the Direct-ARR mode. The Direct mode achieves a PRR that is 0.14 higher than the one observed in the testbed. In general, the other modes perform slightly better in the simulation environment. While Direct-ARR-LOOF still is the mode with the highest PRR, the margin is less pronounced in the simulation environment.

The influence of the x_{EPN} parameter is shown in Fig. 8.19. The results support the statement in Sect. 8.4.1 – $x_{EPN} = 1$ does not have a significant positive influence on the average PRR and therefore the flagging mechanism can be deactivated.

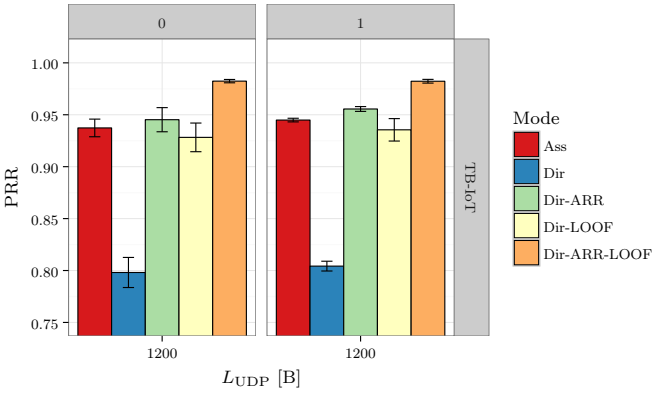


Figure 8.19: PRR of IoTLab testbed for $x_{\mathrm{EPN}} \in [0, 1]$ (columns)

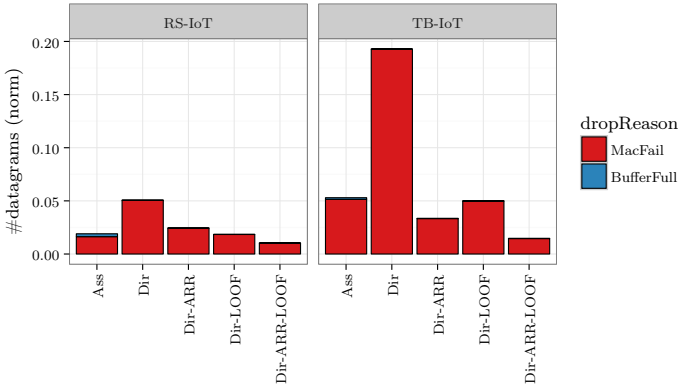
Drop Causes

Looking at the causes for dropped fragments (Fig. 8.20), we can observe that the by far dominating cause for lost fragments are transmission failures at the link layer. Further investigation reveals that among the causes for these drops a consecutively failed CCA and in consequence a failed CSMA/CA is the dominant one. This is true for both simulation and testbed environment and corresponds to the findings in Sect. 7.3.3. The notable differences in simulation and testbed environment for the different forwarding modes can be directly translated to the observations made for the PRR.

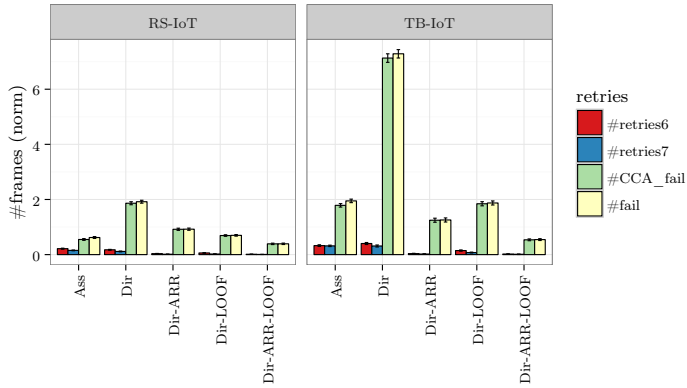
Sniffing

In general, the experiments at the Grenoble site of the IoTLab have been executed at night in a rather quite environment with regard to interference by other wireless technologies. The installed sniffers record a basic noise, which is supposed to be caused by idle beaconing of IEEE 802.11 access points and misinterpretation of transmission belonging to the experiment network itself (cf. Sect. 8.3.1). This "background noise" shows no significant correlation with the results of the experiment. However, with one set of experiments, a more significant interference was recorded.

Figure 8.21 shows a normalized number of registered non-idle sniffer measurements plotted against real-time together with a scatter plot of the average PRR obtained for the Assembly mode. It can be seen that during one of last sets of experiments, the PRR decreases significantly while at the same time the number of non-idle measurements increases significantly. Similar results can be observed for the other forwarding modes (except Direct-ARR-LOOF, whose runs in that period were invalid and removed due to nodes crashing during that particular set of experiments) during the execution of this set of experiments. This result also explains the slightly higher confidence intervals observable in Fig. 8.19, which shows the results of the experiment with significant



(a) Drop causes



(b) IEEE 802.15.4 MAC drop causes

Figure 8.20: Drop causes for fragments of IoTLab testbed and derived simulation scenario

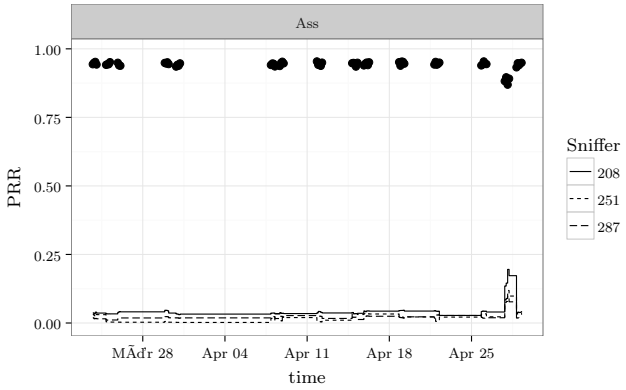


Figure 8.21: Working of the radio sniffer during IoTLab experiment. Scatter plot showing the PRR along with the normalized step plot of the number of sniffer measurements registering unrecognizable signals with an energy in the interval $[-80, -90]$ dBm: $n_{90} \times 10$. Results are shown for all three sniffer nodes (see Fig. 8.10)

sniffer activity. This shows that the installed radio sniffers are viable to detect periods of increased interference by other networks.

Summary

There are several possible approaches to explain significantly better performance of the non-delayed Direct modes in the simulation environment, as observed in Sect. 8.4.2. First, the simulation model may underestimate the busyness on the wireless channel, i.e., the CCA fails more often in the real testbed due an energy detection yielding a result that is larger than the macCcaTh of -90 dBm. However, the data obtained from the radio sniffers show a ratio of busy to total measurements of below 1 % most of the time and therefore rule out this possibility.

Secondly, it is possible that the simulation underestimates the capability of the transceiver to successfully decode the IEEE 802.15.4 SFD of an incoming frame even if the signal is weak. Recall that the used transceiver tries to completely receive a frame before it is ready to receive another one and the used macCcaMode reports a channel as busy if a frame is currently being decoded. Therefore, an increased capability to successfully decode the SFD of weak signals can worsen the overall performance of the testbed. During topology construction, all received frames with a received signal strength of -90 dBm were put into a bin representing RSSI values in the interval $[-90, -100]$ dBm. A link that primarily produces measurements of this uncertain type results in a simulated link with an average RSSI in the interval $[-90, -100]$ dBm. As shown in Fig. 4.3, the simulation model will successfully decode an SFD with a probability near 1, which does not provide much room for underestimation. However, during construction of the topology, only frames received successfully, i.e., with a

λ_B [B s^{-1}]	P_{tx} [dBm]	x_{EPN}
112.5	-3.5	0,1

Table 8.10: Parameter setting for TB-D experiment

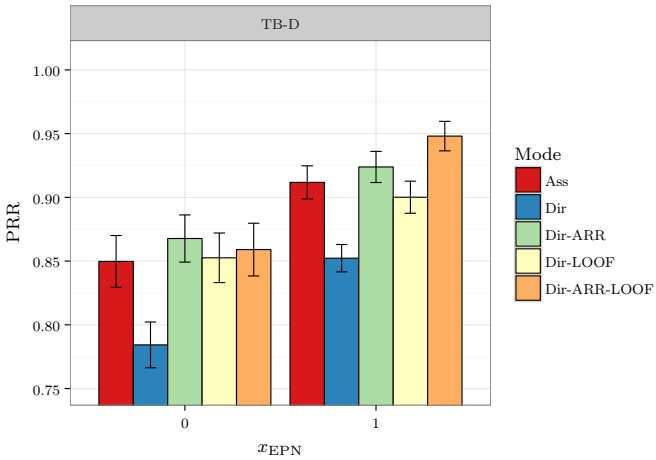


Figure 8.22: PRR in TB-D against x_{EPN}

correct CRC are actually passed to the TopologyMonitor. Frames with a successfully decoded SFD but a failed CRC check are discarded and are not counted. This means that some "SFD-capable" links may exist in the testbed that are not recorded in the corresponding simulation environment and thereby produce the described effect.

8.4.3 TB-D Experiments

Due to the observed fluctuation of link qualities, the experiment in the testbed at Hamburg University of Technology used a slightly increased transmission power P_{tx} compared to the transmission power used to determine the routing topology. The transmission rate of UDP packets in the traffic generator was set to generate a total traffic rate of $\lambda_{B,total} = 4387.5\,\mathrm{B\,s^{-1}}$.

PRR and x_{EPN}

At a first glance, the results from the TB-D contrast those obtained from the IoTLab and simulations.

Figure 8.22 suggests that without using the flagging mechanism (cf. Sect. 8.1.2), the average PRR achieved by the Direct-ARR-LOOF forwarding mode drops by 0.1, along with similarly strong pronounced drops of all other modes. The comparatively large confidence interval also hint at a strong fluctuation in the results for individual runs.

It has to be noted that while the experiments for the non-6LoOF forwarding modes were repeated along with those for the 6LoOF modes, the value of x_{EPN} does not have any influence on the behavior of those modes. Therefore, the x_{EPN} parameter can not be the sole source of influence in this case.

Drop Causes

Examining the number of fragments dropped and the corresponding causes (Fig. 8.23a), it becomes evident that a major part of dropped fragments was caused by the node with id 101. Furthermore, different from earlier observations, a dominant reason for a dropped fragment at the link-layer were consecutively failing retransmissions (and not CSMA/CA failures; Fig. 8.23b). Note that while the execution of the different modes and their repeated runs were interleaved to counter temporarily varying links, the experiments sets for $x_{\mathrm{EPN}} = 1$ and $x_{\mathrm{EPN}} = 0$ had a gap of about two weeks between them. Therefore, an explanation in the observed behavior is, that the link between nodes 101 and 105 was temporarily not available during the experiment with $x_{\mathrm{EPN}} = 0$.

This explanation is further supported by the scatter plot of the PRR averaged over all nodes (Fig. 8.24). There is a strikingly large number of measurements around a PRR of about 0.62 for all forwarding modes. With node 101 being the forwarding router for a part of the network containing 5 out of 13 nodes ($8/13 \approx 0.615$), it is indeed plausible that during the runs achieving a PRR of about 0.62 the link was completely broken. This assumption is further supported by the fact that the recordings of the radio sniffer located at the gateway node show no meaningful correlation with the observed PRR values.

Apart from that, the scatter plots shows that there is a comparatively large variance between individual runs in general for the experiment set with $x_{\mathrm{EPN}} = 0$. This shows that in this case, the measures taken to prevent the static routing topology from working did not prevent all problems and that the temporal variation of link quality has a large if not dominant influence on the results. Therefore, the obtained results are of limited usefulness.

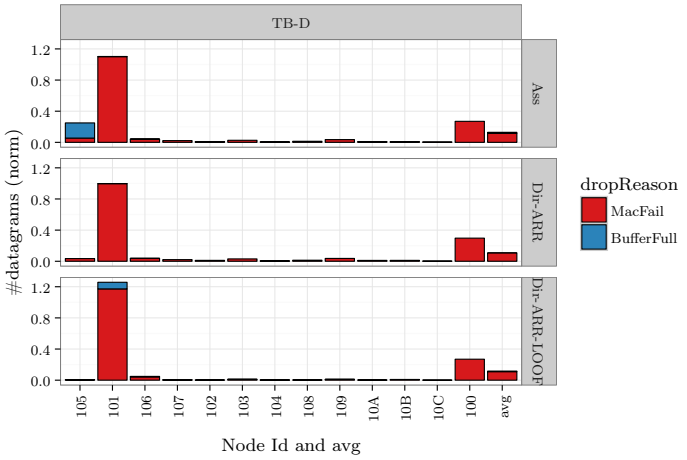
However, the scatter plot also shows that the Direct-ARR-LOOF forwarding mode still exhibits a number of runs with a PRR near 1, which no other mode achieves. This indicates that the Direct-ARR-LOOF mode can outperform the other forwarding modes in the given configuration if all links are present and healthy.

To compensate for the "broken link" effect, the measured data from the testbed was adjusted by removing all runs that exhibited 100 or more fragments at any single node that underwent the maximum number of 7 IEEE 802.15.4 MAC retransmissions.

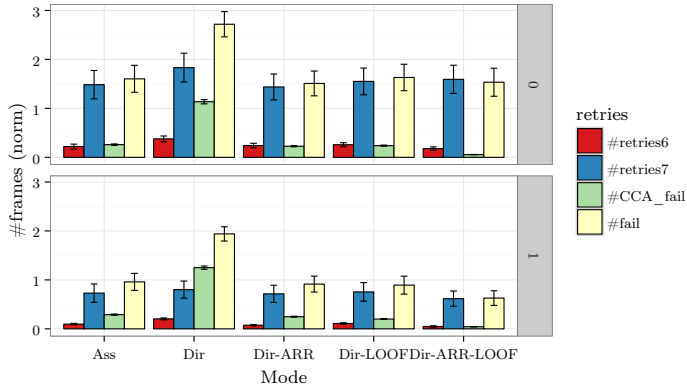
In consequence, the PRR of all forwarding modes is improved for both settings of x_{EPN}. While with $x_{\mathrm{EPN}} = 0$ the PRR on average is still lower compared to $x_{\mathrm{EPN}} = 1$, the relative performance among the forwarding modes is, with the exception of Direct-LOOF and Assembly, much more similar after the removal of the bad runs.

Comparison to Simluation

The comparison of simulation and adjusted testbed data (Fig. 8.25a) leads to results similar to those obtained for the testbed in the IoTLab. The PRR obtained in simulation better by 0.03 (Direct-ARR, Direct-ARR-LOOF) to 0.17 (Direct-LOOF). Again,



(a) Drop causes



(b) IEEE 802.15.4 MAC drop causes

Figure 8.23: Drop causes for fragments of TB-D testbed and derived simulation scenario. Rows represent forwarding mode (8.23a) and x_{EPN} (8.23b)

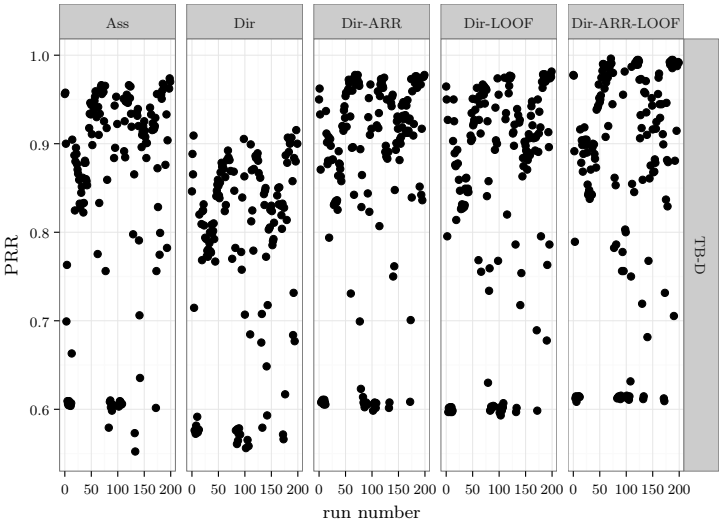


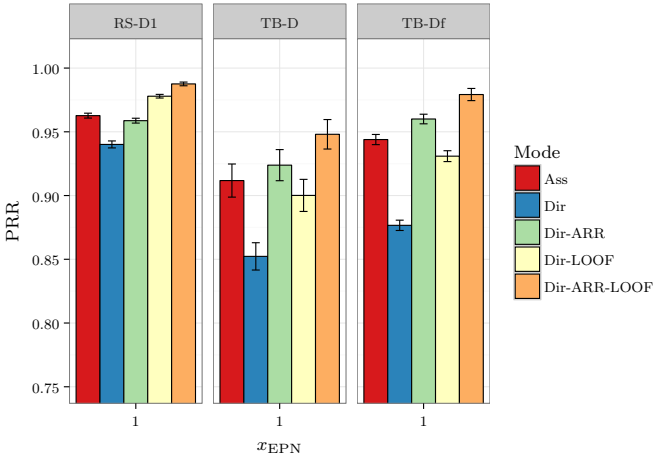
Figure 8.24: Scatter plot of the PRR averaged over all nodes for individual runs for different forwarding modes (columns); x_{EPN}=0

the non-delayed forwarding modes Direct and Direct-LOOF achieve a better PRR in the simulation environment than in the testbed. For the TB-D, however, the main reason for the simulation environment to exhibit a better PRR is the larger number of frames dropped due to consecutively failed retransmissions in the testbed even after the adjustment and not a larger number of CCA failures as can be seen in Fig. 8.25b.

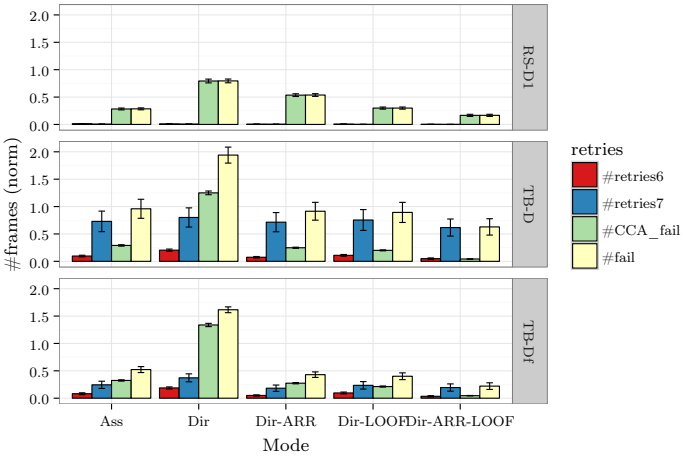
8.4.4 Simulation

The simulation runs iterate over λ_{B}, Q and L_{UDP}. Some of the networks introduced in Sect. 8.3.3 differ with respect to the total number of hops that have to be traversed for the traffic of all nodes to reach its destination. Therefore, the total traffic rate $\lambda_{\mathrm{B,total}}$ would vary significantly if λ_{B} was equal for all networks. To enable a rough comparison between the different topologies, λ_{B} is set to reflect four categories of traffic load T: $\mathrm{T_{low}}$, $\mathrm{T_{mid}}$, $\mathrm{T_{high}}$ and $\mathrm{T_{max}}$.

Table 8.11 shows the setting for each network and the corresponding resulting $\lambda_{\mathrm{B,total}}$ and traffic category. 6LoWPAN buffer space is classified into three categories $\mathrm{Q_S}$, $\mathrm{Q_M}$ and $\mathrm{Q_L}$, corresponding to $Q \in \{2560, 3840, 5120\}$ B for non-6LoOF and to $Q \in \{2304, 3584, 4846\}$ B for 6LoOF forwarding modes. Furthermore, all simulation runs were carried out for different values of the UDP packet size $L_{\mathrm{UDP}} \in \{400, 800, 1200\}$ B.



(a) PRR



(b) IEEE 802.15.4 MAC drop causes

Figure 8.25: PRR and MAC drop causes for TB-D, adjusted TB-Df and derived simulation network

Networks	h_{total}	T_{low}	T_{mid}	T_{high}	T_{max}
RS-C1	57	75.0|4275.0|16	112.5|6412.5|11	150.0|8550.0|8.0	234.4|13360|5.1
RS-D1, RS-D2	39	112.5|4385.5|11	150.0|5850.0|8	187.5|7312.5|6.4	337.5|13163|3.6
RS-IoT	176	28.1|4950.0|43	37.5|6600.0|32	56.3|9900.0|21	75.0|13200|16
Chain-16	120	37.5|4500.0|32	56.3|6750.0|21	75.0|9000.0|16	112.5|13500|11
Y3-6	117	37.5|4387.5|32	56.3|6581.3|21	75.0|8775.0|16	112.5|13163|11
Y6-4	116	37.5|4350.0|32	56.3|6525.0|21	75.0|8700.0|16	112.5|13050|11
Y10-3	111	37.5|4162.5|32	56.3|6243.8|21	75.0|8325.0|16	112.5|12488|11
X3-8	108	37.5|4050.0|32	56.3|6075.0|21	75.0|8100.0|16	112.5|12150|11

Table 8.11: 6LoOF simulation runs; $\lambda_{\text{B}}[\text{B}\,\text{s}^{-1}]|\lambda_{\text{B,total}}[\text{B}\,\text{s}^{-1}]|i[\text{s}]$ for different traffic load categories

Overview of PRR Performance

First, mind that the scale of the y-axis does not begin at zero to better visualize smaller differences. To get an accurate impression, the PRR bar plots have to be regarded as a zoomed-in and cut-off top section of the PRR. Note also that for the $\text{T} = \text{T}_{\text{max}}$ of the testbed-derived networks only the RS-IoT network maintains a PRR above 0.6. The other two networks enter a congested state for that transmission rate

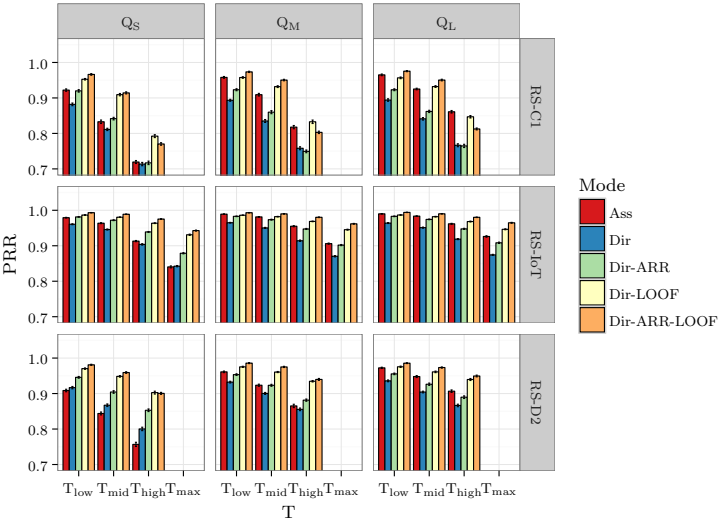


Figure 8.26: PRR against traffic load T for testbed-derived networks (rows), different buffer sizes (columns) and forwarding modes (bar color). The payload was set to $L_{\text{UDP}} = 1200\,\text{B}$

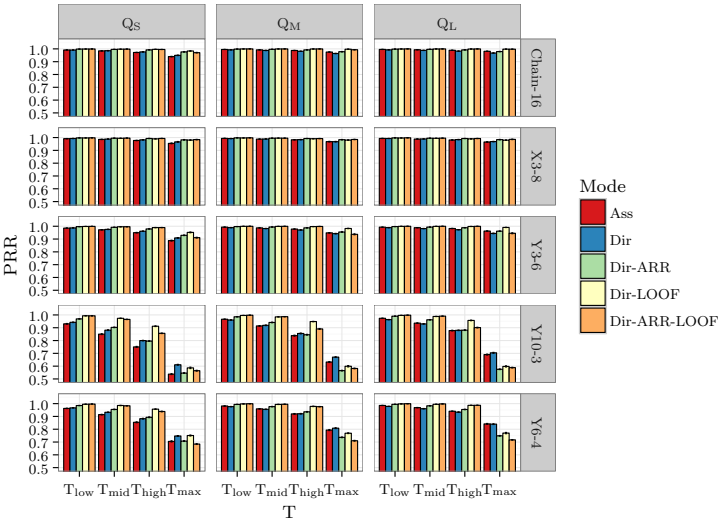


Figure 8.27: PRR against traffic load T for idealized networks (rows), different buffer sizes (cols) and forwarding modes (bar color). The UDP packet payload was set to $L_{\mathrm{UDP}} = 1200\,\mathrm{B}$

and the PRR drops as low as 0.4 for those networks and is deliberately cut off in Fig. 8.26 to retain visibility of differences for the other rates.

Figure 8.26 shows the average PRR for all testbed-derived networks and a UDP payload $L_{\mathrm{UDP}} = 1200\,\mathrm{B}$. For the smallest of the evaluated 6LoWPAN buffer sizes, the 6LoOF mechanism significantly improves the PRR for all networks and all rates in comparison to the Direct-ARR mode, which consistently achieves the best performance of non-6LoOF forwarding modes. With increasing Q, the PRR of the Assembly mode significantly increases, causing it to obtain the best result of the non-6LoOF modes. However, the performance of Direct-ARR-LOOF is still better for all evaluated traffic loads in networks RS-IoT and RS-D1. An exception can be identified with the RS-C1 network, in which the Assembly mode achieves the highest PRR for $\mathrm{T} = \mathrm{T}_{\mathrm{high}}$ and 6LoWPAN buffer size Q_{L}. Another observation is, that in this network, the Direct-LOOF mode achieves a higher PRR than Direct-ARR-LOOF for $\mathrm{T} = \mathrm{T}_{\mathrm{high}}$.

It can also be observed that the PRR of the 6LoOF modes is not significantly increased by switching from a Q_{M} to a Q_{L} buffer. This is similar to the Direct modes. However, even for the Q_{S} buffer, 6LoOF outperforms the plain Direct forwarding modes by 0.03 to 0.07. The gap to the Assembly mode is significantly larger for a Q_{S} buffer, especially for $\mathrm{T} = \mathrm{T}_{\mathrm{high}}$, e.g., 0.1 in network RS-IoT and 0.15 in network RS-D2. This indicates that the goal for 6LoOF to work with restricted buffer space is fulfilled.

The observations for the idealized networks are mainly consistent with those for the testbed-derived ones (Fig. 8.27). Interestingly, while the $\lambda_{B,total}$ is similar to the testbed-derived networks and among the idealized ones, there are significant differences with regard to the PRR-performance. In Chain-16, Y3-6 and X3-8 networks, the PRR achieved by 6LoOF up to $T = T_{high}$ is near 1. This can be attributed to the sharply cut interference range that reaches only the direct neighbor of each node in the idealized topologies. Thereby, contention on the wireless channel is greatly reduced compared to the testbed-derived networks. 6LoOF can be identified as the forwarding mode achieving the best results. Especially in the Y10-3 and Y6-4 networks, 6LoOF exhibits a better PRR for $T = T_{high}$, with the non-rate-restricted Direct-LOOF mode performing better that the rate-restricted Direct-ARR-LOOF. The latter fact is plausible considering the topology of Y10-3 and the limited potential for pipelining fragments in that network.

It has to be noted that approaching congestion in networks Y10-3 and Y6-4 with $T = T_{max}$, the performance of both 6LoOF modes and that of the Direct-ARR mode decrease in comparison to the other forwarding modes. This behavior indicates that 6LoOF is not as well suited in an already congested network. Section 8.1.3 shortly discussed a possible explanation for this behavior: If a successor node has to drop a datagram, a predecessor node will always transmit with snooping rate towards it because it will never get a confirmation for its fragments.

The results of the comparison between the real testbed and derived simulation environments in Sect. 8.4.2 and Sect. 8.4.3 show a significantly higher PRR in simulation especially for the non-rate-restricted Direct modes. This has to be kept in mind when assessing the simulation results.

Drop Causes

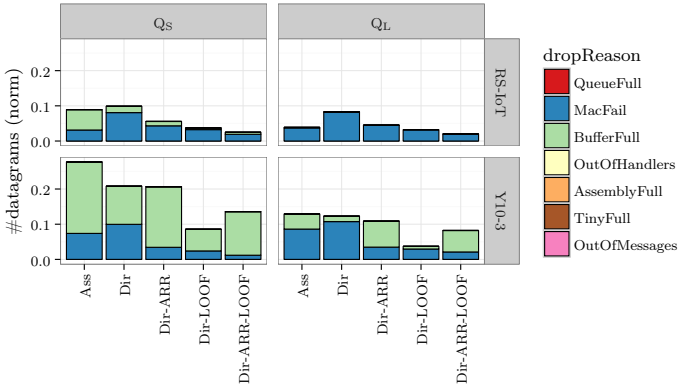
Exemplarily, the causes for dropped fragments are investigated for a 6LoWPAN buffer of size $Q \in \{Q_S, Q_L\}$, $T = T_{high}$ and UDP packet size $L_{UDP} = 1200\,B$ in RS-IoT and Y10-3 networks.

Examining the general causes for dropping fragments (Fig 8.28a) in the RS-IoT network, it becomes evident that a lack of buffer space and failures at the link-layer are the only causes for failures. For the larger 6LoWPAN buffer size, the number of drops due to link-layer transmission failures is increased. This can be explained by a stronger contention for the wireless channel due to less fragments being rejected because of missing buffer space. For the Assembly mode, which has the largest reduction of buffer drops with the larger buffer size, the number of link-layer failures increases more significantly.

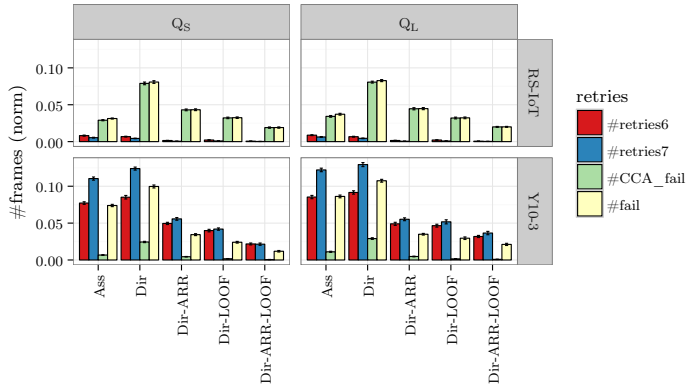
Furthermore, it can be observed that the 6LoOF mechanism effectively reduces both buffer drops and transmission failures. While Direct-LOOF exhibits less buffer drops than Direct-ARR-LOOF, this is more than compensated for by the significantly reduced number of transmission failures.

In RS-IoT, the dominant specific reason for transmission failures can again be identified as CSMA/CA failures (Fig. 8.28b). 6LoOF is not only able to reduce the number of failures caused by consecutively failed retransmissions to near zero, but also exhibits a much smaller number of CSMA/CA failures, i.e., consecutively failed CCAs.

The results are quite different for the idealized Y10-3 network, which contains shorter paths and a single bottleneck at node 2 (Fig. 8.12d). Here, even for the large



(a) General drop causes



(b) IEEE 802.15.4 MAC drop causes

Figure 8.28: Causes for dropping of fragments in general and at the IEEE 802.15.4 link-layer in detail in RS-IoT and Y10-3 networks for $L_{\mathrm{UDP}} = 1200\,\mathrm{B}$

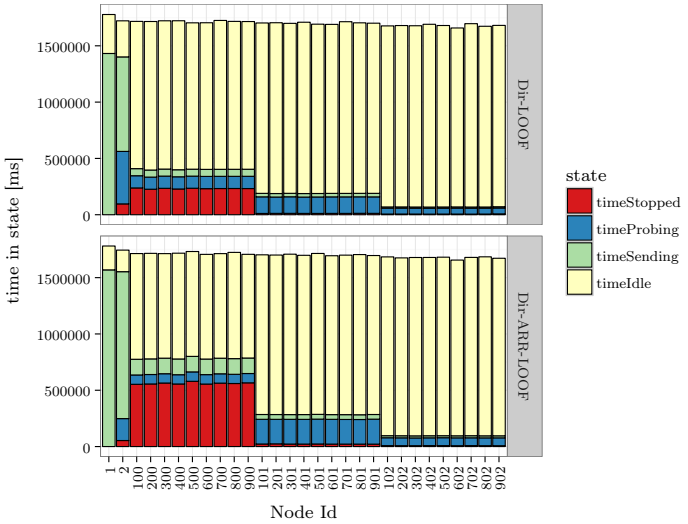


Figure 8.29: Time spent in 6LoOF states against node id for T = T_{high}, Q_L buffer and L_{UDP} = 1200 B as a stacked bar plot. Nodes with an id $X00$ are those just before the bottleneck node on the path.

buffer size, buffer drops represent a large part of the total losses for rate-restricted Direct modes and the Assembly mode. With multiple transmissions being active simultaneously, the rate-restriction mechanism does not work effectively – note that it was mainly constructed to prevent self-interference along a single path. The results even indicate that the additional delay slows down the transmission so that the backlog eventually causes nodes to run out of buffer space.

The cause of link-layer drops also show that the idealized Y10-3 network indeed suffers much less from CSMA/CA failures due to the sharply cut-off interference range of all nodes as assumed in Sect. 8.4.4. In this network, consecutively failed retransmissions are the dominant causes for transmission failures.

6LoOF states

To investigate the inner workings of 6LoOF, the total time spent in each of the states was recorded for each node. As expected, the nodes just before the bottleneck node spent a large part of their non-idle time in state Stopped, whereas nodes situated behind them nearly never stopped but entered probing state comparatively often (Fig. 8.29). This shows that the 6LoOF protocol works as expected: Node 2 has to forward fragments belonging to many different datagrams, causing all its predecessors to suspend their own transmission when they snoop a fragment not belonging to their active datagram.

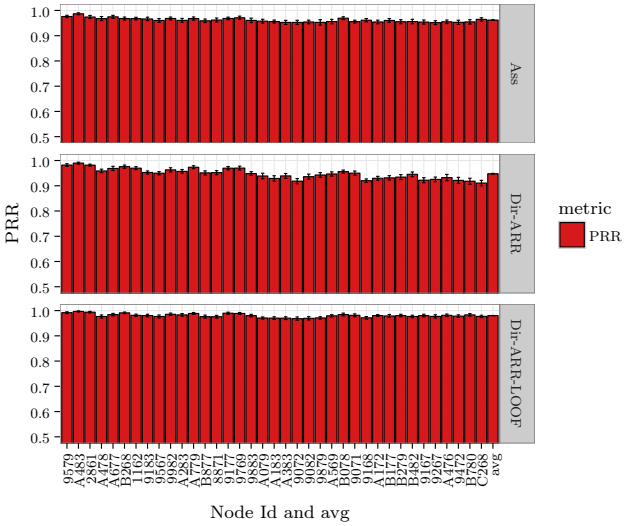


Figure 8.30: Comparison of PRR of individual nodes in RS-IoT network for Assembly, Direct-ARR and Direct-ARR-LOOF modes for $Q = \mathrm{Q_L}$, $\mathrm{T} = \mathrm{T_{high}}$ and $L_{\mathrm{UDP}} = 1200\,\mathrm{B}$. Nodes are ordered in ascending order by their distance (number of hops) to the sink. There is no indication that 6LoOF favors certain nodes.

Compared to the rate-restricted Direct-ARR-LOOF, in Direct-LOOF mode nodes are in state Stopped for significantly less time and also spent less time in state Sending. Both observations indicate that with rate-restriction, nodes complete the transmission of their active datagram less swiftly and eventually have to drop fragments as their buffer runs full. This is confirmed by the drop causes per node that show a much larger number of buffer drops for the rate-restricted modes. Note that state Sending does not only denote the raw time the 6LoWPAN layer request a frame transmission from the link-layer but also incorporates any delays due to rate-restriction.

It can also be seen that the node at the last wireless hop is in state sending for more than 83 % (88 % for Direct-ARR-LOOF) of the duration of the complete experiment. Similar observations can be made for the other evaluated networks with the exception of networks Y3-6 and Chain-16, which exhibit a less busy last-hop node. The node in front of the sink does not employ rate-restriction or 6LoOF suspension for datagrams destined to the sink. Hence, a traffic load of $\mathrm{T} = \mathrm{T_{high}}$ can be equated to a network under heavy load. It has to be noted, however, that a 6LoOF node remains in state Sending while waiting for additional fragments from predecessors.

Fairness

Figure 8.30 shows the PRR for all nodes in the RS-IoT testbed for the forwarding modes Assembly, Direct-ARR and Direct-ARR-LOOF and T = T_{high} and L_{UDP} = 1200 B. Nodes are ordered in ascending order by their distance (number of hops) to the sink. This figure is qualitatively representative for the other evaluated networks. It shows that there is no significant preference to certain nodes by the 6LoOF protocol. While some differences in PRR between individual nodes exist, the pattern is very similar for all the shown forwarding modes and can not be attributed to the 6LoOF protocol. That on average the PRR is slightly decreasing with the distance from the sink is to be expected, considering the increased potential for failures along longer path (cf. Fig. 3.3) and can also not be attributed to the protocol itself. Compared to the plain Direct-ARR forwarding, the absolute differences between nodes are even less pronounced with 6LoOF.

UDPPacket size L_{UDP}

For the Q_L 6LoWPAN buffer and T = T_{high} the performance of the forwarding modes for different UDP packet sizes is evaluated, while keeping the total data rate constant. This is realized by adapting the average interval between packets according to $\lambda = \lambda_B / L_{UDP}$. Note that the PRR is not directly comparable for the different combinations of λ_B and L_{UDP}, because only individual packets are accounted for. On the one hand, the overhead caused by the UDP and IPv6 headers is larger for smaller packets. On the other hand, losing one fragment per 1200 B payload for L_{UDP} = 400 B translates into the loss of one out of three datagrams, whereas for L_{UDP} = 1200 B it translates into losing 1 out of 1 datagram.

Figure 8.31 shows the PRR for $L_{UDP} \in \{400, 800, 1200\}$B for the testbed-derived network topologies, T = T_{high} and a $Q = Q_M$. For L_{UDP} = 400 B in RS-IoT and RS-D2, the PRR for all modes is better for smaller UDP packets. For the smallest shown size of 400 B, the different modes achieve comparable performance in RS-IoT and in RS-D2 and with 6LoOF the decrease is less pronounced for larger UDP packets.

An interesting observation are the distinguished results for the network RS-C1 and Y10-3 for 400 B packets. The nearly exclusive reason for this behavior are drops due to lack of buffer space at node 106. No single dominant reason for the bad performance of the Direct-ARR and both 6LoOF forwarding modes could be identified, but several factors are assumed to contribute to the observed behavior.

First, as described in Chapter 5, the adaptive rate restriction of Direct-ARR and Direct-ARR-LOOF uses an exponentially weighted average over the measured duration for frame transmissions (including backoffs and retransmissions). The median of the frame duration for this scenario is below 10 ms for all nodes. However, Fig. 8.32 shows the measurements classified as outliers (1.5 times the inter-quartile range). Some extremely large values for the duration of a frame transmission can be observed. Even though such values are smoothed by the averaging, they temporarily significantly increase the delay for the next transmissions. With node 106 not being directly connected to the root of the tree, the rate restriction is applied at this single bottle neck to nearly all fragments created in the network.

Secondly, the timeout obtained from the parameter study in Sect. 8.4.1 is not optimal for the smaller packet size. Reducing T_{to} from 5 s to 2 s slightly improves the

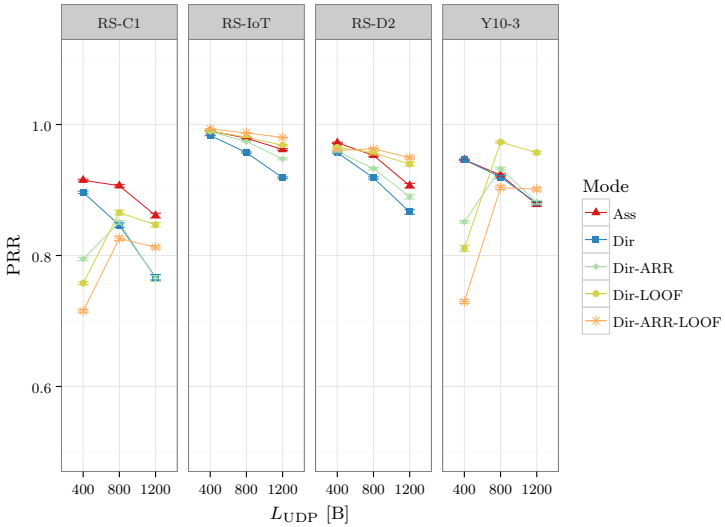


Figure 8.31: PRR against L_{UDP} for testbed-derived networks for $\mathrm{T} = \mathrm{T}_{\mathrm{high}}$ and $Q = \mathrm{Q}_{\mathrm{M}}$. For smaller datagrams, the performance advantage of 6LoOF diminishes or is even inverted.

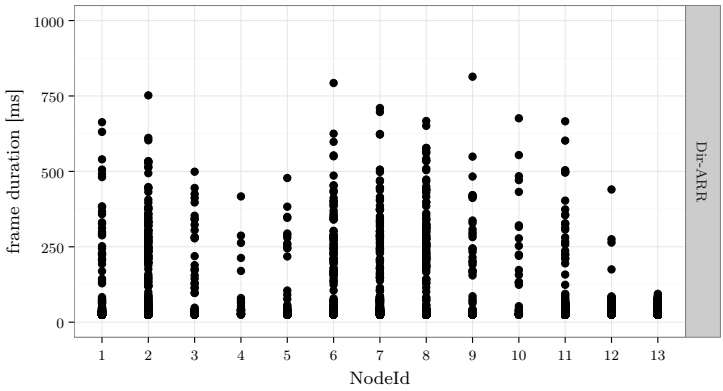


Figure 8.32: Scatter plot of measured duration for frame transmissions larger than 25 ms in ms with Direct-ARR for all nodes in RS-C1. Individual measurements can reach extremely high values.

PRR in this scenario by about 0.02. In the light of this observation, choosing the timeout value for forwarding or reassembly based on the size of the incoming datagram presents an opportunity to improve overall 6LoWPAN performance.

Finally, Fig. 8.31 shows that Direct-LOOF, though being not a rate-restricted forwarding technique, also suffers from a comparatively bad performance for 400 B packets. A possible explanation can be found in the ordering of datagrams. For 400 B packets, the transmission rate at the traffic generator is thrice the rate used for 1200 B packets, resulting in a larger number of independent packets in the network. Therefore, more fragments are transmitted during the initial probing phase of 6LoOF, increasing the overall probability of fragments that do not belong to the active datagram to arrive at a node. Fragments that do belong to the active datagram, on the other hand, can be delayed by those transmissions. If no fragments of the active datagram are available for transmission in such a case, the node remains idle although fragments are ready for transmission until the next fitting fragment arrives or the "push back" mechanism described in Sect. 8.1.3 becomes active.

Polling

The results from randomized periodic collection are compared to a straightforward polling approach, i.e., the sink asks one node after the other to transmit a UDP packet. Thereby, contention on the network is reduced to self-interference between fragments of the same UDP packet. For these simulation runs, the plain forwarding modes were used. In all networks, the polling strategy achieves a PRR near one with forwarding modes Assembly and Direct-ARR (not shown). The PRR of the Direct mode is lower by a few percent but also above 0.95 in all networks. For each requested UDP packet, a timeout of 3 s was used. While this may seem a rather large value, the latency for an individual UDP packet could be observed to occasionally exceed 2 s.

Figure 8.33 shows the average per-node throughput of collecting a single UDP traffic packet from all nodes in the network. As reference, the average throughput achieved by periodic push-based collection of data using Direct-ARR-LOOF forwarding is shown. Comparing the average throughput achieved with polling and and push-based collection for the different networks, we see that for the RS-IoT and X3-8 networks, the throughput is better by about 57 % and 40 %, respectively. In the other networks, polling is outperformed less distinctly (Chain-16, Y3-6, RS-C1) or achieves an even better throughput, albeit with different forwarding modes (Assembly: RS-D1, RS-D2; Direct: Y10-3; Direct-ARR: Y6-4). Achieving a PRR near 1 for $T = T_{max}$, the networks X3-8, Chain-16 and RS-IoT may support higher traffic rates, while the other networks already suffer from congestion for this traffic rate and no further increase can be expected.

The major difference between the RS-IoT and X3-8 networks, which exhibit the best performance compared to polling, and all others is, that apart from the last hop(s), the network can be divided into two spatially divided sub-networks that do not share a common interference range (excluding the last hop towards the gateway node). Thereby, transmissions can truly take place in parallel. This is true especially for RS-IoT because even though the potential for pipelining along the same path is existent, it is limited. This fact can be derived from the only slightly larger throughput achieved by the Direct forwarding modes for polling collection. In this regard, the two

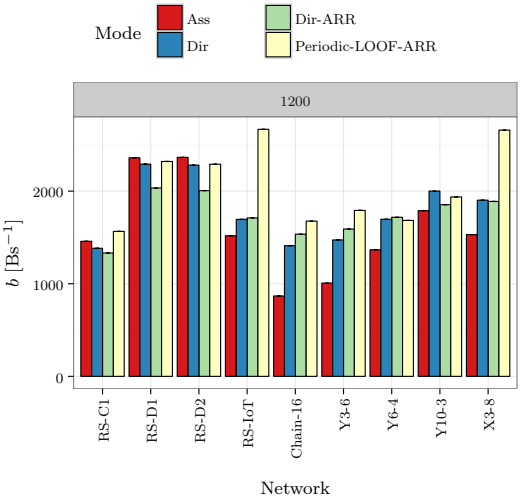


Figure 8.33: Average per-node throughput of data collection for a single round using polling with plain forwarding modes. 1200 B data were requested from each node. For comparison, the highest average per-node throughput achieved by periodic collection is shown for $L_{\mathrm{UDP}} = 1200\,\mathrm{B}$, $Q = \mathrm{Q_M}$; If a PRR of at least 0.9 was achieved the throughput shown is for $\mathrm{T} = \mathrm{T_{max}}$, otherwise for $\mathrm{T} = \mathrm{T_{high}}$.

idealized networks with similar long paths (Chain-16 and Y3-6) can achieve a much higher throughput with the Direct modes compared to the Assembly mode.

8.4.5 Summary

A validation of the simulation model using two distinct testbed networks shows that while not perfect, the simulation model largely exhibits the same tendencies as can be observed in the testbeds. Especially the non-rate-restricted Direct forwarding modes consistently achieve higher PRR results in simulation. A major reason for the difference is the larger part of link-layer transmissions that failed due to consecutive failures in the testbeds in contrast to failures due to failed CSMA/CA mechanisms in the simulation environment.

The evaluation presented in this chapter shows that 6LoOF forwarding does improve the performance of traffic collection in terms of PRR in most of the evaluated collection traffic scenarios. Although 6LoOF's book keeping needs some additional memory, it performs especially well in configurations with small to medium-sized buffers (2560 B, 3840 B). The main reason for the improved PRR is the fact that 6LoOF is able to significantly reduce the number of link-layer failures by putting nodes into state Stopped

and thereby reducing the overall contention on the wireless channel. Exemplarily examining the time spent in each state, it could be seen that nodes effectively suspend their transmissions in favor of other ongoing transmission. These result clearly indicate that the core idea behind the 6LoOF mechanism works well for large datagrams. Furthermore, the results indicate that 6LoOF is at least as fair as the plain Direct forwarding modes.

However, there are some drawbacks. Maintaining the same average traffic rate but reducing the payload of individual UDP packets, the advantage that 6LoOF achieves for very large datagrams diminishes. In some topologies, an significant decrease of the PRR compared to the other forwarding strategies could be observed for a packet size of 400 B in combination with a traffic rate near the congestion state of the network. Also, 6LoOF performs slightly worse than the other forwarding modes if the network is in a congested state.

One of the unsolved problems of 6LoOF is the issue of "snoop-starving" described in Sect. 8.1.3. Apart from explicitly notifying the predecessors, which necessitates changes to the 6LoWPAN mechanism, no potential solution to this issue can be presented.

The evaluation also shows that polling is a viable alternative to periodic randomized push-based data collection for large datagrams. Especially in networks with no or limited potential for parallel data transmissions, a simple polling scheme achieves the same or even slightly better performance than periodically triggered data collection. If, however, potential for parallelization is existent, be it via pipelining on the same path or segregation of the network into multiple near-independent parts, push-based data collection outperforms polling in terms of achievable throughput, which can be also translated into a larger number of supportable nodes in the network.

9 Conclusion and Outlook

The ever increasing demand for ubiquitous autonomously networked sensing and control systems, also known as the "Internet of Things" or "Cyber-Physical Systems" constitutes a number of new challenges. One of those is the question of how to integrate large wireless networks of cheap and resource-constrained devices into existing network infrastructures. A proposed solution is to take the proven and time-tested family of internet protocols and make them usable on even the tiniest nodes. Several protocols that are built upon the internet protocol version 6 have been standardized by the IETF. Among them, the 6LoWPAN protocol addresses issues of IPv6 header compression and fragmentation for link-layers that are not capable of providing the full minimum MTU of 1280 B.

This dissertation addresses the issues that are caused by fragmentation of large datagrams and their transmission via multi-hop routes using the IEEE 802.15.4 link and physical layer. It concentrates on data collection traffic scenarios, i.e., multiple nodes, which periodically transmit data to a sink, that utilize the unslotted CSMA/CA mode of the IEEE 802.15.4 MAC protocol. While some research on this topic in form of analytical models and simulative and experimental evaluation exists, the available work either makes strong simplifications or evaluates the different forwarding strategies using unrealistically small networks. Furthermore, the influence of the configuration of underlying link-layer is usually regarded not further. Another proposed approach to improving 6LoWPAN reliability was based on end-to-end negative acknowledgments and can be seen as orthogonal to the research presented here. Taking the efforts further, this dissertation makes the following contributions to the problem.

An analytical model based on bit-error probabilities was extended to better reflect practical approaches to handle the forwarding of fragmented datagrams. By means of the extended model, the number of "useless" bits, i.e., data belonging to datagrams that have already been lost, can be quantified when using a non-reassembling approach to fragment forwarding. The output of the model suggests that for a reasonably high overall end-to-end datagram success rate, the impact of these already-lost fragments is limited. While neglecting important aspects of wireless communication, the model was also used to show the strong negative influence the combination of fragmentation and multi-hop transmissions have on the end-to-end datagram success rate.

A suitable simulation model for investigating the performance of 6LoWPAN fragmentation was discussed. Additionally, a method to derive a network topology including the link-layer properties from an existing testbed was developed and implemented for the CometOS framework, which can be used as a basic operating system for wireless sensor nodes and within the OMNeT++ simulator, using MiXiM to provide the capability of sophisticated modeling of wireless physical channels. By comparing results from the original testbed and a corresponding testbed-derived simulation environment, a validation of the simulation model could be achieved and thereby the confidence in the results from the more large-scale simulations could be improved.

With initial experiments and simulations, the two most basic route-over forwarding modes Assembly and Direct were evaluated together with a forwarding mode which adaptively restricts the rate of transmissions to prevent self-interference. The results showed a dramatically low PRR especially for the examined Direct forwarding strategies, whereas the Assembly mode produced results consistent with expectations. Comparing these results to the ones obtained with another model for the bit-error probability of the IEEE 802.15.4 layer clearly indicated, that either the simulation did not generate realistic results or some flaw in the testbed existed. A property of the transceiver's "extended operating mode", which provides support for automatic CSMA/CA and retransmissions, is that the transceiver becomes incapable of receiving any incoming IEEE 802.15.4 frame as soon as an automatic transmission is initiated. This "no-RX-while-TX" property was finally suspected to be a potential cause for the non-consistent results.

In consequence, an experiment setup was devised and implemented to quantify the actual influence of the "no-RX-while-TX" property of the transceiver. For this, the detailed sequence of states of the used radio stack for each of the four participating nodes was recorded during the transmission of a large datagram along a path of three hops. As a reference, the same experiments were repeated with the radio driver layer of TinyOS (ported to CometOS) which implements CSMA/CA backoffs and retransmissions in software and does allow for the reception of frames during the backoff phase of a transmission. The obtained results clearly show that the "no-RX-while-TX" property indeed was solely responsible for the observed results. It has to be noted that this finding can be important for all other testbed deployments which use the same transceiver or a transceiver with a similar mechanism and IEEE 802.15.4's unslotted CSMA/CA mode. Even though the impact is especially strong in multi-hop scenarios with fragmentation, it has potential to bias all results taken from deployments using this combination. An example for this are M3 OpenNodes running RIOT OS, which is a probable setup to be encountered in the large and widely-used testbed facility IoTLab.

After throwing out the transceiver's extended operating mode, a new parameter study was set up with the aim to assess the influence and a suitable set of IEEE 802.15.4 parameters for 6LoWPAN fragmentation of large datagrams. Furthermore, the performance of the different forwarding modes was evaluated for varying parameter sets, varying backoff exponents, the maximum number of retransmissions, the 6LoWPAN buffer size and the mode of clear channel assessments IEEE 802.15.4 uses. The findings of the parameter study include that a larger number of maximum link-layer retransmissions for all evaluated scenarios improve the overall reliability. Additionally, the IEEE 802.15.4 minimum backoff exponent has a significant influence on the end-to-end datagram success rate. The mode of IEEE 802.15.4 channel assessment does have controversial influence, as the more aggressive variant (mode 3) improved the PRR in a network with artificially increased transmission power but significantly decreased it for the same routing topology with non-increased transmission power. Discussion of the results led to a selection of parameters to be used for the evaluation of the 6LoOF protocol.

Another finding of the evaluation was that a simple polling scheme for certain topologies proved to be a viable and easy alternative to periodic collection of large data items triggered by the nodes independently. In some networks, the polling approach

achieves comparable throughput and in nearly all cases a better PRR. The evaluation indicates that polling is viable for large data packets, if the potential for pipelining is limited and no parallel sub-networks exist in the network.

Finally, this dissertation proposes 6LoOF, an implementation of 6LoWPAN that uses snooping of network traffic to order the transmission of datagrams with the aim to reduces the overall contention for the wireless channel and thereby improve the overall reliability of the transmission of large, 6LoWPAN-fragmented datagrams. The basic idea is that if a node overhears the transmission of a datagram that is not the datagram currently transmitted by the node itself, the node will suspend its own transmission to enable fast completion of the overheard datagram. This also implies an "ordering" of fragments according to their belonging to a certain datagram, which is reflected in the name of the mechanism. Mechanisms to prevent node starving and suspension deadlocks were discussed and some additional tweaks to the protocol presented to improve its performance. 6LoOF was implemented in for CometOS and evaluated in two different testbeds, at the IoTLab and at Hamburg University of Technology. Furthermore, a simulation study was carried out, comparing 6LoOF to other forwarding modes in testbed-derived and idealized networks.

The evaluation showed that 6LoOF outperforms the other forwarding modes in nearly all of the evaluated scenarios. It consistently was able to reduce the number of failures at the link-layer that otherwise led to the loss of a whole datagram. Especially with regard to the usage of buffer space, 6LoOF shows that it utilizes the available space better than the Assembly mode and works well with small to medium buffer sizes. Disabling the optional use of an explicit probing flag by means of a modified header dispatch value, 6LoOF can be implemented completely compliant to the 6LoWPAN standard, sacrificing only a tiny part of the reliability improvement.

There are, however, some limitations of the 6LoOF protocol. First, it does not offer a significant improvement when smaller datagrams are forwarded. In some cases, especially those of a high traffic load near or in a congested network state, it even performs worse for smaller datagrams than the other forwarding modes. Adapting the forwarding/reassembly timeout for smaller fragments may mitigate this problem, but not completely. Secondly, one inherent problem of the protocol is its inability for a node to communicate to a predecessor that the datagram belonging to an incoming fragment has been discarded due to a link-layer failure or the lack of buffer space. In consequence, predecessor nodes have to transmit the remaining fragments in a probing state, wasting much time with an already lost cause. A possible remedy to this situation is the transmission of explicit notifications every time an invalid fragment is detected. This approach, however, could not be evaluated in this dissertation, because the issue and a possible solution were discovered too late in the process of completion. Finally, it was found that 6LoOF does not perform as well in scenarios with a very high traffic load. The aforementioned issue is likely to be one of the reasons for this.

Applied together with one of the end-to-end congestion control mechanism that are proposed for CoAP to alleviate its shortcomings, the 6LoOF mechanism can be used to improve the overall reliability in application scenarios that demand for the periodic collection of large data items using a completely standardized protocol stack.

Bibliography

[06] *IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)*. Sept. 2006.

[10] *IEC 62591:2010 - Industrial communication networks - Wireless communication network and communication profiles - WirelessHART^{TM}*. IEC, 2010.

[11a] *IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*. Sept. 2011.

[11b] *User Manual Radio Modules; deRFmega128-22A00, deRFmega128-22A02, deRFmega128-22C00, deRFmega128-22C02*. Document Version V1.4. dresden electronic. Dresden, Germany, Aug. 2011.

[12a] *IEEE 802.15.4e^{TM}-2012 -IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer*. IEEE, 2012.

[12b] *ZigBee Specification*. 2400 Camino Ramon, Suite 375, San Ramon, CA 94583: ZigBee Alliance, Sept. 2012.

[14] *8-bit AVR Microcontroller with Low Power 2.4GHz Transceiver for ZigBee and IEEE 802.15.4: ATmega256RFR2*. Rev. C. Atmel Corporation. San José, Sept. 2014.

[15] *Low-Power, 2.4 GHz ISM-Band IEEE 802.15.4 RF Transceiver with Extended Proprietary Features: MRF24XA*. Rev. C. Microchip Technology Inc. Chandler, Arizona, Apr. 2015.

[16] *IEEE 802.15.4^{TM}-2015 - IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (WPANs)*. IEEE, 2016.

[Adj+15] Cedric Adjih et al. "FIT IoT-LAB: A large scale open experimental IoT testbed". In: *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*. Dec. 2015, pp. 459–464.

[Ali+06] Muneeb Ali, Umar Saif, Adam Dunkels, Thiemo Voigt, Kay Römer, Koen Langendoen, Joseph Polastre, and Zartash Afzal Uzmi. "Medium Access Control Issues in Sensor Networks". In: *SIGCOMM Comput. Commun. Rev.* 36.2 (Apr. 2006), pp. 33–36.

[AMR11] Ahmed Ayadi, Patrick Maille, and David Ros. "TCP over Low-Power and Lossy Networks: Tuning the Segment Size to Minimize Energy Consumption". In: *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*. Feb. 2011, pp. 1–5.

[Aya+11] Ahmed Ayadi, Patrick Maille, David Ros, Laurent Toutain, and Pascal Thubert. "Energy-efficient fragment recovery techniques for Low-Power and Lossy Networks". In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*. July 2011, pp. 601–606.

[Bac+12] Nouha Baccour, Anis Koubâa, Luca Mottola, Marco Antonio Zúñiga, Habib Youssef, Carlo Alberto Boano, and Mário Alves. "Radio Link Quality Estimation in Wireless Sensor Networks: A Survey". In: *ACM Trans. Sen. Netw.* 8.4 (Sept. 2012), 34:1–34:33.

[Bac+13] Emmanuel Baccelli, Oliver Hahm, Mesut Günes, Matthias Wählisch, and Thorsten C. Schmidt. "RIOT OS: Towards an OS for the Internet of Things". In: *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*. Apr. 2013, pp. 79–80.

[Bea96] David M. Beazley. "SWIG: An Easy to Use Tool for Integrating Scripting Languages with C and C++". In: *Proceedings of the 4th Conference on USENIX Tcl/Tk Workshop, 1996 - Volume 4*. TCLTK'96. Monterey, California: USENIX Association, 1996, pp. 15–15.

[Bet+15] August Betzler, Carles Gomez, Ilker Demirkol, and Josep Paradells. "CoCoA+: An advanced congestion control mechanism for CoAP". In: *Ad Hoc Networks* 33 (2015), pp. 126–139.

[BGD15] August Betzler, Carles Gomez, and Ilker Demirkol. "Evaluation of Advanced Congestion Control Mechanisms for Unreliable CoAP Communications". In: *Proceedings of the 12th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*. PE-WASUN '15. Cancun, Mexico: ACM, 2015, pp. 63–70.

[Bia06] Giuseppe Bianchi. "Performance Analysis of the IEEE 802.11 Distributed Coordination Function". In: *IEEE J.Sel. A. Commun.* 18.3 (Sept. 2006), pp. 535–547.

[Boc+11] Maurizio Bocca, Janne Toivola, Lasse M. Eriksson, Jakko Hollmén, and Heiko Koivo. "Structural Health Monitoring in Wireless Sensor Networks by the Embedded Goertzel Algorithm". In: *Cyber-Physical Systems (ICCPS), 2011 IEEE/ACM International Conference on*. Apr. 2011, pp. 206–214.

[BS16] Carsten Bormann and Zach Shelby. *Block-wise transfers in CoAP (draft-ietf-core-block-20)*. `https://tools.ietf.org/html/draft-ietf-core-block-20`. accessed: 2016-05-11. Apr. 2016.

[Bue+06] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. "X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks". In: *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. SenSys '06. Boulder, Colorado, USA: ACM, 2006, pp. 307–320.

[Cho+09] Aminul Haque Chowdhury, Muhammad Ikram, Hyon-Soo Cha, Hassen Redwan, S. M. Saif Shams, Ki-Hyung Kim, and Seung-Wha Yoo. "Route-over vs mesh-under routing in 6LoWPAN". In: *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing Connecting the World Wirelessly (IWCMC)*. 2009, pp. 1208–1212.

[CMN14] Cosmin Cobârzan, Julien Montavont, and Thomas Noël. "Analysis and performance evaluation of RPL under mobility". In: *2014 IEEE Symposium on Computers and Communications (ISCC)*. June 2014, pp. 1–6.

[DC12] Stephen Dawson-Haggerty and David E. Culler. *BLIP Tutorial*. `http://tinyos.stanford.edu/tinyos-wiki/index.php/BLIP_Tutorial`. accessed: 2016-05-24. 2012.

[De +03] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. "A High-throughput Path Metric for Multi-hop Wireless Routing". In: *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*. MobiCom '03. San Diego, CA, USA: ACM, 2003, pp. 134–146.

[DGV04] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. "Contiki - a lightweight and flexible operating system for tiny networked sensors". In: *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. Nov. 2004, pp. 455–462.

[Di +12] Piergiuseppe Di Marco, Pangun Park, Carlo Fischione, and Karl Henrik Johansson. "Analytical Modeling of Multi-hop IEEE 802.15.4 Networks". In: *Vehicular Technology, IEEE Transactions on* 61.7 (Sept. 2012), pp. 3191–3208.

[Dij] Edsger W. Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1.1 (), pp. 269–271.

[Duq+15] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH". In: *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. SenSys '15. Seoul, South Korea: ACM, 2015, pp. 337–350.

[Fam+14] Olivier Fambon, Eric Fleury, Gaëtan Harter, Roger Pissard-Gibollet, and Frédéric Saint-Marcel. ""FIT IoT-LAB Tutorial: Hands-on Practice With a Very Large Scale Testbed Tool for the Internet of Things"". In: *10èmes journées francophones Mobilité et Ubiquité, UbiMob* (2014).

[Gna+09] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. "Collection tree protocol". In: *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*. Berkeley, California: ACM, 2009, pp. 1–14.

[GS84] Clark R. Givens and Rae Michael Shortt. "A class of Wasserstein metrics for probability distributions." In: *Michigan Math. J.* 31.2 (1984), pp. 231–240.

[Hau09] Jan-Hinrich Hauer. *TKN15.4: An IEEE 802.15.4 MAC Implementation for TinyOS 2*. TKN Technical Report Series TKN-08-003. Telecommunication Networks Group, Technical University Berlin, Mar. 2009.

[HC08] Jonathan W. Hui and David E. Culler. "Extending IP to Low-Power, Wireless Personal Area Networks". In: *IEEE Internet Computing* 12.4 (July 2008), pp. 37–45.

[HC11] Ulrich Herberg and Thomas Clausen. "A Comparative Performance Study of the Routing Protocols LOAD and RPL with Bi-directional Traffic in Low-power and Lossy Networks (LLN)". In: *Proceedings of the 8th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*. PE-WASUN '11. Miami, Florida, USA: ACM, 2011, pp. 73–80.

[HT11] Jonathan Hui and Pascal Thubert. *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*. RFC 6282. Sept. 2011.

[Ise+15] Javier Isern, August Betzler, Carles Gomez, Ilker Demirkol, and Josep Paradells. "Large-Scale Performance Evaluation of the IETF Internet of Things Protocol Suite for Smart City Solutions". In: *Proceedings of the 12th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*. PE-WASUN '15. Cancun, Mexico: ACM, 2015, pp. 77–84.

[JDK15] Ilpo Järvinen, Laila Daniel, and Markku Kojo. "Experimental evaluation of alternative congestion control algorithms for Constrained Application Protocol (CoAP)". In: *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*. Dec. 2015, pp. 453–458.

[Joh94] Glenn E. Johnson. "Constructions of particular random processes". In: *Proceedings of the IEEE* 82.2 (Feb. 1994), pp. 270–285.

[KG14] Hamidreza Kermajani and Carles Gomez. "On the Network Convergence Process in RPL over IEEE 802.15.4 Multihop Networks: Improvement and Trade-Offs". In: *Sensors (Basel, Switzerland)* 14.7 (July 2014), pp. 11993–12022.

[Kim+07a] Sukun Kim, Rodrigo Fonseca, Prabal Dutta, Arsalan Tavakoli, David Culler, Philip Levis, Scott Shenker, and Ion Stoica. "Flush: A Reliable Bulk Transport Protocol for Multihop Wireless Networks". In: *Proceedings of the 5th international conference on Embedded networked sensor systems*. SenSys'07. New York, NY, and USA: ACM, 2007, pp. 351–365.

[Kim+07b] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steven Glaser, and Martin Turon. "Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks". In: *2007 6th International Symposium on Information Processing in Sensor Networks*. Apr. 2007, pp. 254–263.

[Ko+10] Teresa Ko, Shaun Ahmadian, John Hicks, Mohammad Rahimi, Deborah Estrin, Stefano Soatto, Sharon Coe, and Michael P. Hamilton. "Heartbeat of a Nest: Using Imagers As Biological Sensors". In: *ACM Trans. Sen. Netw.* 6.3 (June 2010), 19:1–19:31.

[Köp+08] Andreas Köpke et al. "Simulating Wireless and Mobile Networks in OMNeT++ the MiXiM Vision". In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Simutools '08. Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, 71:1–71:8.

[Kur+12] Takuto Kuroiwa, Makoto Suzuki, Yasutaka Yamashita, Shunsuke Saruwatari, Tomonori Nagayama, and Hiroyuki Morikawa. "A multi-channel bulk data collection for structural health monitoring using wireless sensor networks". In: *2012 18th Asia-Pacific Conference on Communications (APCC)*. Oct. 2012, pp. 295–299.

[LCC11] Alessandro Ludovici, Anna Calveras, and Jordi Casademont. "Forwarding Techniques for IP Fragmented Packets in a Real 6LoWPAN Network". In: *Sensors* 11.1 (2011), pp. 992–1008.

[LCL07] HyungJune Lee, A. Cerpa, and P. Levis. "Improving Wireless Simulation Through Noise Modeling". In: *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*. 2007, pp. 21–30.

[Lev+05] Philip Levis et al. "Ambient Intelligence". In: ed. by Werner Weber, Jan M. Rabaey, and Emile Aarts. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. Chap. TinyOS: An Operating System for Sensor Networks, pp. 115–148.

[Lev+11] Philip Levis, Thomas Clausen, Jonathan Hui, Omprakash Gnawali, and JeongGil Ko. *The Trickle Algorithm*. RFC 6206. Mar. 2011.

[Lud+14] Alessandro Ludovici, Piergiuseppe Di Marco, Anna Calveras, and Karl H. Johansson. "Analytical Model of Large Data Transactions in CoAP Networks". In: *Sensors* 14.8 (2014), pp. 15610–15638.

[MLT08] Razvan Musaloiu-E., Chieh-Jan Mike Liang, and Andreas Terzis. "Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks". In: *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*. IPSN '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 421–432.

[Mon+07] Gabriel Montenegro, Nandakishore Kushalnagar, Jonathan Hui, and David Culler. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. RFC 4944. Sept. 2007.

[MT15] Florian Meier and Volker Turau. "An analytical model for fast and verifiable assessment of large scale wireless mesh networks". In: *Design of Reliable Communication Networks (DRCN), 2015 11th International Conference on the*. Mar. 2015, pp. 185–190.

[Nar+07] Thomas Narten, Erik Nordmark, W. Simpson, and Hesham Soliman. *Neighbor Discovery for IP version 6 (IPv6)*. RFC 4944. Sept. 2007.

[NH93] Homayoun Nikookar and Homayoun Hashemi. “Statistical modeling of signal amplitude fading of indoor radio propagation channels”. In: *Universal Personal Communications, 1993. Personal Communications: Gateway to the 21st Century. Conference Record., 2nd International Conference on*. Vol. 1. Oct. 1993, 84–88 vol.1.

[Pae+05] Jeongyeup Paek, Krishna Chintalapudi, John Caffrey, Ramesh Govindan, and Sami Masri. “A Wireless Sensor Network for Structural Health Monitoring: Performance and Experience”. In: (2005).

[PG10] Jeongyeup Paek and Ramesh Govindan. “RCRT: Rate-Controlled Reliable Transport Protocol for Wireless Sensor Networks”. In: *ACM Transactions on Sensor Networks (TOSN)* 7.3 (2010).

[Pos80] John Postel. *User Datagram Protocol*. RFC 768. Aug. 1980.

[PTB10] Dimosthenis Pediaditakis, Yuri Tselishchev, and Athanassios Boulis. “Performance and Scalability Evaluation of the Castalia Wireless Sensor Network Simulator”. In: *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*. SIMUTools ’10. Torremolinos, Malaga, Spain: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010, 53:1–53:6.

[Puc+10] Daniele Puccinelli, Omprakash Gnawali, SunHee Yoon, Silvia Giordano, and Leonidas Guibas. “END: A Topology-aware Collection Metric for Sensor Networks”. In: *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. SenSys ’10. Zürich, Switzerland: ACM, 2010, pp. 419–420.

[Rap02] Theodore S. Rappaport. *Wireless communications : principles and practice*. 2. ed. Prentice Hall communications engineering and emerging technologies series. Upper Saddle River, NJ: Prentice Hall PTR, 2002.

[Ren13] Bernd-Christian Renner. “Sustained Operation of Sensor Nodes with Energy Harvesters and Supercapacitors”. PhD thesis. Hamburg, Germany: Hamburg University of Technology, 2013.

[RH10] George F. Riley and Thomas R. Henderson. “The ns-3 Network Simulator”. In: *Modeling and Tools for Network Simulation*. Ed. by Klaus Wehrle, Mesut Güneş, and James Gross. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34.

[RL08] Tal Rusak and Philip A. Levis. “Investigating a Physically-based Signal Power Model for Robust Low Power Wireless Link Simulation”. In: *Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM ’08)*. Vancouver, British Columbia, Canada: ACM, 2008, pp. 37–46.

[Sey05] John S. 1958- Seybold. *Introduction to RF propagation*. Wiley-Interscience, 2005.

[SHB14] Zach Shelby, Klaus Hartke, and Carsten Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. June 2014.

[SK11] Sujesha Sudevalayam and Purushottam Kulkarni. “Energy Harvesting Sensor Nodes: Survey and Implications”. In: *IEEE Communications Surveys Tutorials* 13.3 (Third 2011), pp. 443–461.

[SKH06] Dongjin Son, Bhaskar Krishnamachari, and John Heidemann. “Experimental Study of Concurrent Transmission in Wireless Sensor Networks”. In: *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. SenSys ’06. Boulder, Colorado, USA: ACM, 2006, pp. 237–250.

[Sri+10] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. “An Empirical Study of Low-power Wireless”. In: *ACM Trans. Sen. Netw.* 6.2 (Mar. 2010), 16:1–16:49.

[Suz+07] Makoto Suzuki, Shunsuke Saruwatari, Narito Kurata, and Hiroyuki Morikawa. “A High-density Earthquake Monitoring System Using Wireless Sensor Networks”. In: *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*. SenSys ’07. Sydney, Australia: ACM, 2007, pp. 373–374.

[TB12] Emanuele Toscano and Lucia Lo Bello. “Comparative assessments of IEEE 802.15.4/ZigBee and 6LoWPAN for low-power industrial WSNs in realistic scenarios”. In: *Factory Communication Systems (WFCS), 2012 9th IEEE International Workshop on*. May 2012, pp. 115–124.

[TH14] Pascal Thubert and Jonathan Hui. *LLN Fragment Forwarding and Recovery (draft)*. `https://tools.ietf.org/html/draft-thubert-6lo-forwarding-fragments-01`. accessed: 2014-09-10. Feb. 2014.

[Thu15] Pascal Thubert. *An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4*. `http://tools.ietf.org/pdf/draft-ietf-6tisch-architecture-09.pdf`. accessed: 2016-05-19. Nov. 2015.

[TOV10] Joydeep Tripathi, Jaudelice C. de Oliveira, and Jean-Philippe Vasseur. “A performance evaluation study of RPL: Routing Protocol for Low power and Lossy Networks”. In: *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*. Mar. 2010, pp. 1–6.

[Unt14] Stefan Unterschütz. “Methodologies and Protocols for Wireless Communication in Large-Scale, Dense Mesh Networks”. PhD thesis. Hamburg, Germany: Hamburg University of Technology, 2014.

[UWT12] Stefan Unterschütz, Andreas Weigel, and Volker Turau. “Cross-Platform Protocol Development Based on OMNeT++”. In: *Proceedings of the 5th International Workshop on OMNeT++ (OMNeT++’12)*. Desenzano, Italy, Mar. 2012.

[Var+01] András Varga et al. “The OMNeT++ discrete event simulation system”. In: *Proceedings of the European simulation multiconference (ESM’2001)*. Vol. 9. S 185. sn. 2001, p. 65.

[Var99] Andras Varga. “Using the OMNeT++ discrete event simulation system in education”. In: *IEEE Transactions on Education* 42.4 (Nov. 1999), p. 372.

[VH08] András Varga and Rudolf Hornig. "An Overview of the OMNeT++ Simulation Environment". In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Simutools '08. Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, 60:1–60:10.

[Wat+16] Thomas Watteyne, Vlado Handziski, Xavier Vilajosana, Simon Duquennoy, Oliver Hahm, Emmanuel Baccelli, and Adam Wolisz. "Industrial Wireless IP-Based Cyber-Physical Systems". In: *Proceedings of the IEEE* 104.5 (May 2016), pp. 1025–1038.

[Wei+14a] Andreas Weigel, Christian Renner, Volker Turau, and Holger Ernst. "Wireless Sensor Networks for Smart Metering". In: *Energy Conference and Exhibition (ENERGYCON), 2014 IEEE International*. Dubrovnik, Croatia, May 2014, pp. 722–729.

[Wei+14b] Andreas Weigel, Martin Ringwelski, Volker Turau, and Andreas Timm-Giel. "Route-over forwarding techniques in a 6LoWPAN". In: *EAI Endorsed Transactions on Mobile Communications and Applications* 14.5 (Dec. 2014).

[Wei10] Andreas Weigel. "Adaptive Channel Selection in Multi-Gateway Wireless Sensor Networks". MA thesis. Am Schwarzenberg-Campus 3, D-21073 Hamburg: Hamburg University of Technology, Aug. 2010.

[Wer+06] Geoffrey Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. "Fidelity and Yield in a Volcano Monitoring Sensor Network". In: *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*. OSDI '06. Seattle, Washington: USENIX Association, 2006, pp. 381–396.

[Wes+09] Karl Wessel, Michael Swigulski, Andreas Köpke, and Daniel Willkomm. "MiXiM - The Physical Layer: An Architecture Overview". In: *Proceeding of the 2. International Workshop on OMNeT++*. Rome, Italy, Mar. 2009.

[WGG10] Klaus Wehrle, Mesut Günes, and James Gross, eds. *Modeling and Tools for Network Simulation*. Springer, 2010.

[Win+12] Tim Winter et al. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC 6550. Mar. 2012.

[WPG15] Thomas Watteyne, Maria Rita Palattella, and Luigi Alfredo Grieco. *Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement*. RFC 7554. May 2015.

[WT14] Andreas Weigel and Volker Turau. "An Analytical Model of 6LoWPAN Route-Over Forwarding Practices". In: *Conference proceedings of the International Conference on Ad-hoc, Mobile and Wireless Networks, ADHOC-NOW 2014*. Benidorm, Spain, June 2014, pp. 279–289.

[WT15] Andreas Weigel and Volker Turau. “Hardware-Assisted IEEE 802.15.4 Transmissions and Why to Avoid Them”. In: *Conference proceedings of the 8th International Conference on Internet and Distributed Computer Systems, IDCS 2015*. Windsor, UK, Sept. 2015, pp. 223–234.

[WV16] Qin Wang and Xavier Vilajosana. *6top Protocol (6P)*. `http://tools.ietf.org/pdf/draft-ietf-6tisch-6top-protocol-00.pdf`. accessed: 2016-05-19. Apr. 2016.

[YCI13] Jiazi Yi, Thomas Clausen, and Yuichi Igarashi. “Evaluation of routing protocol for low power and Lossy Networks: LOADng and RPL”. In: *Wireless Sensor (ICWISE), 2013 IEEE Conference on*. Dec. 2013, pp. 19–24.

[Zhu+13] Yi-Hua Zhu, Gan Chen, Kaikai Chi, and Yanjun Li. “The Chained Mesh-Under Routing (C-MUR) for Improving IPv6 Packet Arrival Rate over Wireless Sensor Networks”. In: *Advances in Wireless Sensor Networks*. Vol. 334. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2013, pp. 734–743.

[ZK04] Marco Zuniga and Bhaskar Krishnamachari. “Analyzing the transitional region in low power wireless links”. In: *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*. Oct. 2004, pp. 517–526.

Glossary

ARQ automatic repeat request

AWGN additive white Gaussian noise

BEP bit error probability

BFSK binary frequency shift keying

CCA clear channel assessment

CoAP Constrained Application Protocol

CSMA/CA carrier sense multiple access with collision avoidance

DCF decider correction factor

DSME distributed synchronous multi-channel extension to IEEE 802.15.4

DSSS direct sequence spread-spectrum

EWMA exponentially weighted moving average

FCF frame control field of the IEEE 802.15.4 MAC header

IEEE 802.15.4 IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks

HTTP Hypertext Transfer Protocol

IETF Internet Engineering Task Force

IPHC 6LoWPAN IPHC header compression

IPv6 Internet Protocol version 6

LFFR LLN fragment forwarding and recovery

LLN low power, lossy network

6LoOF 6LoWPAN ordered forwarding

6LoWPAN abbreviation for the protocol defined by RFC4944 “Transmission of IPv6 Packets over IEEE 802.15.4 Networks”

macCcaMode IEEE 802.15.4 CCA mode (four different variants are specified)

macMinBe IEEE 802.15.4 minimum backoff exponent (unslotted CSMA)

MTU maximum transmission unit

NHC 6LoWPAN next header compression

FEP frame error probability

PRC progress-based retry control

PRR packet success rate

PSDU PHY service data unit

FSP frame success probability

RPL Routing Protocol for Low Power and Lossy Networks

RSSI received signal strength indicator

SFD start of frame delimiter

SINR signal to noise and interference ratio

SNR signal to noise ratio

TCP Transmission Control Protocol

6tisch IPv6 over the TSCH mode of IEEE 802.15.4e

UDP User Datagram Protocol

WSN wireless sensor network

List of Symbols

α smoothing factor of EWMA to estimate T_{tx}

$BE_{\mathrm{6LoOF,max}}$ maximum backoff exponent of 6LoOF probing mechanism

$BE_{\mathrm{6LoOF,min}}$ minimum backoff exponent of 6LoOF probing mechanism

BE_{max} maximum backoff exponent of unslotted IEEE 802.15.4 CSMA/CA

BE_{min} minimum backoff exponent of unslotted IEEE 802.15.4 CSMA/CA

c number of correctable bit errors when using forward error correction

E^{A} expectation value of number of bits sent (Assembly)

E^{D} expectation value of number of bits sent (Direct)

$E^{\mathrm{A}}_{\mathrm{f}}$ expectation value of number of bits sent in case of failure (Assembly)

$E^{\mathrm{A}}_{\mathrm{s}}$ expectation value of number of bits sent in case of success (Assembly)

$H_{\mathrm{f},k}$ expectation value of number of bits sent in case of failure on link k

$H_{\mathrm{p},k}$ expectation value of number of bits sent in case of partial failure on link k

$H_{\mathrm{s},k}$ expectation value of number of bits sent in case of success on link k

$H_{\mathrm{sp},k}$ expectation value of number of bits sent in case of success or partial failure on link k

h index of last hop of a route

h_0 index of first hop of a route

h_{total} accumulated number of hops of all routes in the network participating in the given collection traffic pattern

I random part of interval between UDP packet transmissions in traffic generator

i interval between UDP packet transmissions in traffic generator

i_{max} maximum interval between UDP packet transmissions in traffic generator

λ average rate of generation of UDP packets in traffic generator

λ_B per-node payload data rate in traffic generator

$\lambda_{B,total}$ total raw offered data rate, obtained by the sum of the length of all traffic paths in the network

L_A link layer acknowledgment frame size (bits)

L_F link layer frame size (bits), including 802.15.4 headers

l A single path (route) participating in the collection traffic pattern

$\mathcal{L}$ Set of all paths used for collection traffic

L_{UDP} payload size of UDP packets in traffic generator

m total number of fragments to transmit

N_0 thermal noise power

$n_{f,80}$ number of sniffer samples with energy above −80 dBm and a foreign IEEE 802.15.4 signal being decoded

$n_{f,90}$ number of sniffer samples with energy between −90 dBm and −80 dBm and a foreign IEEE 802.15.4 signal being decoded

$n_{f,idle}$ number of sniffer samples with energy below −90 dBm and a foreign IEEE 802.15.4 signal being decoded

n_{80} number of sniffer samples with energy above −80 dBm and no IEEE 802.15.4 signal

n_{90} number of sniffer samples with energy between −90 dBm and −80 dBm and no IEEE 802.15.4 signal

n_{idle} number of sniffer samples with energy below −90 dBm

$n_{NSWC,max}$ value of nswc that triggers a node to enter state ProbingBo

$n_{o,80}$ number of sniffer samples with energy above −80 dBm and an internal IEEE 802.15.4 signal being decoded

$n_{o,90}$ number of sniffer samples with energy between −90 dBm and −80 dBm and an internal IEEE 802.15.4 signal being decoded

$n_{o,idle}$ number of sniffer samples with energy below −90 dBm and an internal IEEE 802.15.4 signal being decoded

n_{qsa} "queue switch after" – number of empty queue object events 6LoOF after which 6LoOF switches to the next object in the 6LoWPAN transmission queue

p_e bit error probability

$p_{e,\text{frame}}$ packet error probability

$p_{\text{e},k}$ bit error probability on link k

$P_{\text{f},k}$ probability of failure after r attempts on link k

$p_{\text{f},k}$ probability of failure for a single transmission on link k

P_{tx} transmission power of transceiver

$P_{\text{p},k}$ probability of partial failure after r attempts on link k

$p_{\text{p},k}$ probability of partial failure for a single transmission on link k

$\mathcal{P}_{\text{s}}$ end-to-end probability of success for a whole datagram

$P_{\text{s},k}$ probability of success after r attempts on link k

$p_{\text{s},k}$ probability of success for a single transmission on link k

Q_{L} "large" 6LoWPAN buffer; Q=5120 B or Q=4864 B, depending on forwarding mode

Q_{M} "midsize" 6LoWPAN buffer; Q=3840 B or Q=3584 B, depending on forwarding mode

Q_{S} "small" 6LoWPAN buffer; Q=2560 B or Q=2304 B, depending on forwarding mode

Q size of the 6LoWPAN buffer

r maximum number of IEEE 802.15.4 retransmissions

N_{to} timeout counter of aging timeout mechanism

I_{to} timeout interval of aging timeout mechanism

T_{to} timeout value of aging timeout mechanism

T categorized traffic load to make different λ_{B} comparable for different networks

T_{low} traffic category low

T_{mid} traffic category mid

$\mathrm{T}_{\mathrm{high}}$ traffic category high

$\mathrm{T}_{\mathrm{max}}$ traffic category highest

T_{tx} average duration of link layer transmissions, including CSMA/CA

T_{d} delay between link layer transmissions

x_{EPN} flag indicating if the explicit probing notification flag, along with a special 6LoWPAN header type is used in 6LoOF