Analytical Traffic Model of 6TiSCH using Real-Time In-Band Telemetry

Dries Van Leemput, Jeroen Hoebeke, Eli De Poorter

IDLab, Department of Information Technology, Ghent University - imec, Technologiepark-Zwijnaarde 126, 9000 Ghent, Belgium e-mail: firstname.lastname@ugent.be.

Preprint submitted to Elsevier Internet of Things - DOI: https://doi.org/10.1016/j.iot.2023.100847

Analytical Traffic Model of 6TiSCH using Real-Time In-Band Telemetry

Dries Van Leemput, Jeroen Hoebeke, Eli De Poorter

IDLab, Department of Information Technology, Ghent University - imec, Technologiepark-Zwijnaarde 126, 9000 Ghent, Belgium e-mail: firstname.lastname@ugent.be.

Abstract

In industrial environments, the IETF 6TiSCH stack is a protocol stack consisting of standardized layers (IEEE 802.15.4, TSCH, 6LoWPAN, RPL, CoAP) that enable reliable wireless communications. To efficiently manage the network, gathering insights is critical. This allows for detecting traffic bottlenecks, predicting energy consumption, checking duty cycle limits, adjusting network topology, and identifying interference. As such, detailed knowledge of the traffic quantity can be used to create a digital twin of the network to help prevent node or network failures by re-configuring network parameters. To that end, this paper presents an analytical model to calculate the transmitted and received bytes for every node in a 6TiSCH network, considering the complete 6TiSCH network stack. Real-time in-band network telemetry is used to limit monitoring overhead by piggybacking information onto transmitted data packets. Simulations demonstrate that our model provides accurate predictions with a median of 95 to 99% under various circumstances, such as topology changes and packet loss. The model can enhance network robustness and can be extended to predict the energy consumption of 6TiSCH devices easily.

Keywords: IETF 6TiSCH, Traffic Prediction, In-Band Network Telemetry, TSCH, RPL, 6LoWPAN, CoAP.

1. Introduction

Industry 4.0 aims to improve industrial applications and optimize processes by introducing a higher degree of digitalization, which enables new

Preprint submitted to Elsevier Internet of Things - DOI: https://doi.org/10.1016/j.iot.2023.100847

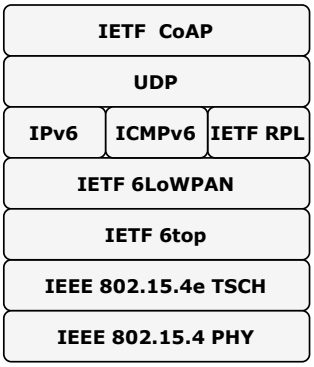


Figure 1: The 6TiSCH network stack consists of multiple protocol layers: the IEEE 802.15.4 PHY and MAC layer, the IETF 6top and 6LoWPAN adaptation layers, IPv6 and ICMPv6, the IETF Routing Protocol for Low-power and lossy networks (RPL), UDP, and the Constrained Application Protocol (CoAP).

insights and technologies such as machine learning, artificial intelligence, digital twins, etc. The Industrial Internet of Things (IIoT) is integral to achieving these goals as it allows continuous measurement from numerous wireless sensors. However, Wireless Sensor Networks (WSNs) must be scalable, reliable, low latency, and energy-efficient, despite having constraints like limited memory, processing power, and battery life. Additionally, harsh industrial environments typically experience signal reflections, multi-path propagation, and fading. To address these challenges, the Internet Engineering Task Force (IETF) IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) working group has proposed the 6TiSCH protocol stack (Fig. 1). This protocol stack consists of multiple standardized protocol layers that provide reliable and deterministic behavior, making it possible to integrate constrained devices into Internet Protocol version 6 (IPv6) networks.

In realistic multi-hop scenarios, packet loss, traffic congestion, and battery depletion can occur frequently, making it crucial for network managers to have current topology and network traffic information to make informed decisions regarding network (re)configuration. Ideally, this allows a central manager to detect and resolve imminent network congestion, breaches in duty-cycle regulations, battery depletion, etc. before occurring. To that end, existing solutions aim to predict how often sensor devices generate data [1]. However, the 6TiSCH network stack itself also introduces a non-negligible amount of traffic due to control messages, packet headers, multi-hop routing, etc. Therefore, we present an analytical model to predict 6TiSCH network

traffic that considers the complete network stack.

Due to the constraints of the network nodes, the calculations should be performed at a central network manager, which requires topology and timing information to be available. To obtain this information, all intermediary nodes insert network telemetry in existing frames, which introduces very limited overhead and provides real-time telemetry to the central manager. This also ensures that the model is able to react to changes in network topology, such as nodes joining or leaving the network. In summary, the main contributions of this work are:

- We present, to the best of our knowledge, the first analytical model of a full 6TiSCH network, while considering all protocols of the 6TiSCH network stack (TSCH, RPL, 6LoWPAN, and CoAP).
- In contrast to existing theoretical models, we also propose a method to efficiently collect the necessary network information and make it available to a central manager that can solve the analytical model in real-time without prior network information.
- We extend an existing In-band Network Telemetry solution (used for efficiently collecting network data from WSNs) to also enable data collection when using the storing mode of RPL.
- We analyse the overhead for collecting telemetry data to achieve different accuracy levels for conditions with and without packet loss.
- Finally, we provide open-source of the model and telemetry approach [2].

The remainder of this paper is structured as follows. Section 2 describes the required technical background on 6TiSCH, and Section 3 gives an overview of related works. Next, Section 4 explains the operation and different sub-models of the model, and defines certain assumptions. Subsequently, Sections 5, 6, and 7 describe the single-hop, multi-hop, and control traffic overhead sub-models in more detail. Then, Section 8 details the in-band network telemetry solution and Section 9 evaluates the proposed model through simulations. Finally, Section 10 concludes the paper.

2. Technical Background

The IEEE 802.15.4 standard [3] defines multiple PHYsical layers (PHYs) and Medium Access Control (MAC) layers targeted at low-rate wireless networks with limited battery consumption requirements. Time-Slotted Channel Hopping (TSCH) combines Time Division Multiplexing (TDM) with Frequency Division Multiplexing (FDM) to provide a MAC suitable for industrial environments due to its high reliability, low duty cycle, and resilience against frequency selective fading. Time is divided into timeslots, which are organized into repeating slotframes. A single timeslot allows for the transmission of a frame and an optional ACKnowledgement (ACK), can be either dedicated to a single transmitter-receiver pair or can be shared between multiple devices. Advertisement timeslots are used to transmit Enhanced Beacons (EBs), which nodes use to synchronize and advertise the network. A channel offset is assigned to every timeslot, which, combined with the absolute slot number, results in channel hopping to combat frequency selective fading.

To enable IPv6 communication using low-power networks, the IETF 6TiSCH working group defined a set of protocols based on the TSCH MAC mode of IEEE 802.15.4, depicted in Fig. 1. Two adaptation layers were proposed in between the MAC layer and higher layers: 6Tisch OPeration sublayer (6top) and IPv6 over Low-Power Personal Area Networks (6LoWPAN). The 6top adaptation layer executes one or multiple Scheduling Functions (SFs), which define when to add or remove cells from the TSCH schedule, and terminates the 6top Protocol (6P) to let nodes communicate removals or additions of cells. As there is no default SF specified in the IEEE 802.15.4 standard, the implementation of the SF is entirely up to the developer. Common SFs include Orchestra [4] and Minimal Scheduling Function (MSF) (RFC9033). The 6LoWPAN adaptation layer allows the use of IPv6 on top of IEEE 802.15.4 as IPv6 requires packet sizes larger than the Maximum Transmission Unit (MTU) of IEEE 802.15.4. To that end, 6LoWPAN specifies a compression format for network headers in addition to the possibility of fragmenting the frame payload over multiple fragments.

6TiSCH uses the Routing Protocol for Low-power and lossy networks (RPL) as it is seen as the de-facto routing protocol for WSNs. Nodes in RPL are organized in a Destination Oriented Directed Acyclic Graph (DODAG) structure with one node acting as the RPL root. Each node is assigned a rank, which increases as the node resides further down the DODAG, and is com-

puted by an implementation-dependent Objective Function (OF). Two possible OFs are Objective Function zero (OF0) and Minimum Rank with Hysteresis Objective Function (MRHOF), described in RFC6552 and RFC6719, respectively. RPL defines two Mode of OPerations (MOPs): non-storing mode and storing mode. In non-storing mode, nodes do not keep a routing table and all traffic is therefore routed towards the root, which inserts a routing header to forward the frame to its destination. In contrast, nodes in storing mode do keep a routing table, which reduces the overall packet overhead but increases memory consumption on the nodes. In addition to RPL, IPv6 and Internet Control Message Protocol version 6 (ICMPv6) complete the network layer, which sits below the User Data Protocol (UDP) as transport layer.

Finally, the Constrained Application Protocol (CoAP) is used as application protocol because of its features for constrained networks and since it enables low-overhead, secure, RESTful interaction.

3. Related work

This section describes related work focusing on (i) prior existing analytical models for 6TiSCH and (ii) state-of-the-art solutions for efficiently collecting network information.

3.1. Analytical 6TiSCH Network Models

Due to its deterministic nature, 6TiSCH is often modelled in literature. Table 1 lists other 6TiSCH models and indicates which network protocols are included. As can be seen, most models capture a single protocol, while only a few works include multiple. In contrast, to the best of our knowledge, we present the first model covering the complete 6TiSCH network stack, including TSCH, RPL, 6LoWPAN, and CoAP. As our model is applicable to a general 6TiSCH network, we do not consider 6top since its use is dependent on the SF and implementation specific.

Some works focus on the behaviour of the TSCH MAC by modelling the energy consumption [5–9], reliability [7–11], latency [7–10], network formation [12], etc. However, none of these works model the overhead introduced by TSCH, and some focus on shared cells or network formation only. In contrast, we model the overhead while considering a network in a steady-state, and include both shared and dedicated cells.

Related works covering RPL also model different characteristics of the network. The convergence time and network formation process of RPL was modelled in [13, 14], whereas [15–17] focus on the Trickle Algorithm. Our model differs from the former works as we do not consider network formation, but focus on a steady-state network. While [15, 16] do study steady-state networks, we also incorporate additional control traffic.

In contrast to the works modelling TSCH or RPL, several more recent works include both protocols in their model. Again, a frequently modelled characteristic of TSCH-RPL networks is network formation [18–21]. Next to network formation, the authors in [22] estimate the energy consumption by modelling the occurrence of different TSCH slot types. 6top is also included in the model presented in [23].

In terms of 6LoWPAN and CoAP models, the authors in [24] model the number of bits sent in the presence of route-over forwarding and 6LoWPAN fragmentation, whereas the congestion in 6LoWPAN is modelled in [25], including the number of received packets at the final destination. The authors in [26] relate the CoAP application packet loss to the network packet loss. Finally, both 6LoWPAN and CoAP are incorporated in [27], as 6LoWPAN fragmenting and CoAP block-wise transfer are compared.

Table 1: Comparison of 6TiSCH network models: in contrast to other works, our work models the complete network stack and incorporate Real-Time Monitoring (RTM) of the network.

Related	**TSCH**	**RPL**	**6top**	**6LoWPAN**	**CoAP**	**RTM**
[5–12]	✓	✗	✗	✗	✗	✗
[13–17]	✗	✓	✗	✗	✗	✗
[18–22]	✓	✓	✗	✗	✗	✗
[23]	✓	✓	✓	✗	✗	✗
[24, 25]	✗	✗	✗	✓	✗	✗
[26]	✗	✗	✗	✗	✓	✗
[27]	✗	✗	✗	✓	✓	✗
[28–31]	✗	✗	✗	✗	✗	✓
Our work	✓	✓	✗	✓	✓	✓

Next to incorporating the complete 6TiSCH network stack, our modelling of the individual network protocols in 6TiSCH also differs from the litera-

ture models presented above, as prior scientific publications rarely model the network traffic, but focus instead on modelling energy consumption, network convergence, collision probability, etc. In addition, these models often rely on (unrealistic) prior network knowledge such as known topologies, and validate their work using offline datasets. In contrast, as shown in the right column of Table 1, we collect network information using non-intrusive network monitoring and validate our model in real-time.

3.2. Network Telemetry in 6TiSCH

This section describes approaches that have been proposed to provide real-time monitoring in 6TiSCH networks.

The authors of [30] present a light-weight telemetry solution for monitoring 6TiSCH [31], based on the Alternate Marking Performance Monitoring (AMPM) concept. While the overhead is close to zero as only a single bit is used for telemetry collection, only end-to-end and hop-by-hop reliability and delay can be monitored, which is insufficient for creating advanced network models. Lahmadi et al. [28] propose a monitoring framework for Low-power and Lossy Networks (LLNs) that minimizes overhead while still being adaptive to topology changes. By exploiting available routing data, the authors assign poller roles to certain nodes and piggyback monitoring data and poller assignment information embedded in standard packets. Their algorithm was validated in simulation and a real-life 6LoWPAN deployment. Gaillard et al. [29] present a mechanism to collect end-to-end delivery ratio and delay for clients and/or applications in 6TiSCH by piggybacking Information Elements (IEs) onto dedicated (additional) control packets. Similarly, Karaagac et al. [30] present a solution for 6TiSCH monitoring using IEs to piggyback information through the network. Instead of using dedicated control packets, the IEs are added to data frames to create a flexible and powerful in-band network telemetry solution to support a wide range of monitoring operations and use cases. Due to its flexibility and low communication and energy consumption overhead, this telemetry solution will form the basis for real-time telemetry collection in our work. However, the authors only consider the non-storing mode of RPL. In contrast, in this paper, we extend their solution to also include storing mode and use DAO packets instead of data packets. As such, our solution is capable of piggybacking network information on control traffic that needs to be transmitted anyway.

4. Overview of Analytical Model and Assumptions

The goal of our analytical model is to predict the number of transmitted and received bytes by every node in a 6TiSCH network. Fig. 2 shows an overview of the analytical models and the different required components to implement the prediction model into a real-life 6TiSCH network. To limit computation and energy consumption overhead on the nodes, the model is executed on a Computation Unit (CU), connected to the network 6LoWPAN Border Router (6LBR). Therefore, as mentioned in Section 3, we employ the In-band Network Telemetry (INT) strategy proposed in [30] to provide network telemetry from the nodes to the 6LBR, which forwards this information to the CU. The telemetry comprises topology and timing information and is encapsulated in IEs, added to data frames, to limit overhead. This eliminates the need for a manual initialization phase because all necessary information can be obtained through INT. Therefore, the model is applicable to various network configurations, irrespective of the network topology, and is able to react to network changes due to the real-time telemetry. Next to telemetry, the model takes data traffic information (payload, source, destination, and timestamp) as input and calculates the introduced overhead by the network. This way, the number of TX and RX bytes can be calculated for every node.

The model is divided in three sub-models, each modelling a different type of overhead: control traffic overhead, single-hop overhead, and multi-hop overhead. The control traffic sub-model calculates the number of transmitted and received control frames for the TSCH, RPL, and ICMPv6 protocols, using timing and topology information gathered with INT. The single-hop overhead sub-model calculates the number of additional bytes due to headers and fragmentation for both data and control traffic. Subsequently, the multi-hop overhead considers the forwarding of frames in case sender and receiver are not within one-hop distance, using the available topology information. Calculating single-hop overhead and multi-hop overhead for both data and control traffic allows us to calculate the total number of transmitted and received bytes for every device in the network.

Before delving deeper into the details of the model, we define four assumptions for the 6TiSCH network and the analytical model:

A 1 (**Known data traffic**)**.** Since our model takes data traffic information as input, we assume this to be known in advance for each node, i.e., the payload, destination, and timestamp of every data frame. Predicting data traffic is out of scope for this paper, as suitable prediction models are available

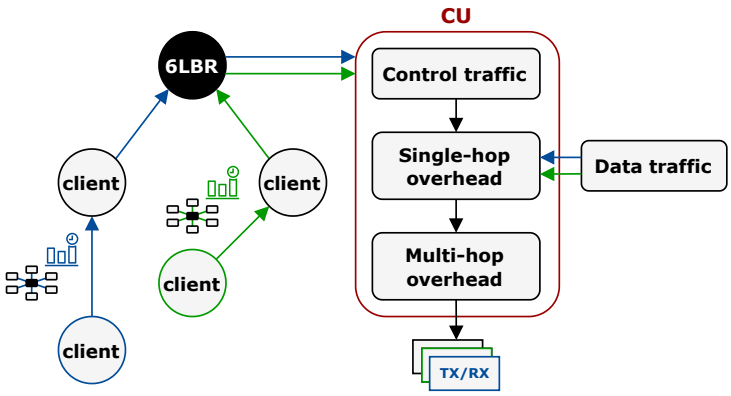


Figure 2: Overview of the analytical model. The Computation Unit (CU) calculates the number of received and transmitted bytes for every node in the network, using three sub-models to determine the control traffic, single-hop, and multi-hop overhead. Data traffic is assumed to be known and the 6LoWPAN Border Router (6LBR) collects network telemetry from the individual nodes.

in literature. Our model aims to extend these models by calculating the associated network overhead. Moreover, nodes could use INT to provide history of their own traffic to the CU, or supply estimations of future traffic requirements.

A 2 (**Limited topology changes**)**.** We assume the network has been deployed and converged but do support infrequent topology changes such as adding, removing, or moving nodes. We do not focus on initial network formation because (i) this is a short period compared to the complete network lifetime and (ii) this is already covered in previous works [18–21].

A 3 (**LoWPAN-bound asymmetric traffic**)**.** All data traffic is assumed to be bound within a single Low-power Wireless Personal Area Network (LoWPAN), where the bulk of the information flows from a source to a destination (asymmetric traffic). The source can either be a client pushing data to a server, or a server publishing data to a client.

A 4 (**Unique DODAG**)**.** We assume a network with one unique DODAG, where the single 6LBR operates as RPL root and TSCH coordinator.

The following sections discuss each sub-model in more detail, starting with single-hop overhead in Section 5, multi-hop overhead in Section 6, and control traffic overhead in Section 7.

5. Single-Hop Overhead

The single-hop overhead sub-model takes CoAP payload as input and calculates all transmitted bytes by the source and destination. We distinguish three types of single-hop overhead in a 6TiSCH network: fixed network headers inherent to the 6TiSCH frame format, CoAP block transfer, and 6LoWPAN fragmenting. We start with examining the standard 6TiSCH packet format, including all possible headers, before deriving a model for both CoAP block transfer, and 6LoWPAN fragmenting.

5.1. 6TiSCH Frame Format

Fig. 3 depicts the frame format for a standard 6TiSCH data frame, split into an IEEE 802.15.4 frame (top), a 6LoWPAN frame (middle) and a CoAP frame (bottom right). The IEEE 802.15.4 frame format includes both PHY and MAC headers. The PHY Service Data Unit (PSDU) is preceded by a Synchronization HeadeR (SHR) and PHY HeadeR (PHR), of which the length is variable and depends on the chosen PHY. The PSDU comprises the MAC Service Data Unit (MSDU), multiple MAC headers and a MAC footer. The size of these headers is configurable and defined in the Frame Control field. The standard specification defines multiple addressing modes, for intra-Personal Area Network (PAN) and inter-PAN communication with either short (2 B) or extended (8 B) addresses. Due to *A 3*, we only consider intra-PAN communication, which results in the source PAN ID being elided. For security concerns, an optional Auxiliary Security Header (ASH) may be included to specify the security configuration of the frame. IEEE 802.15.4 enables the use of IEs to transport information between nodes, of which the authors in [32] list the IEs applicable to 6TiSCH. Usually, IEs are used exclusively by 6TiSCH control plane frames, but since INT also employs IEs, they can also be present in data frames. Because the size of the headers is configurable, we assume this configuration to be known by the CU for every frame type. However, the Addressing Fields and IEs are still variable depending on the frame type, e.g., the source address field in EBs is elided, while it is carried inline for data frames. Therefore, a fixed SHR and PHR can be assumed, but the MAC header size depends on the frame type.

The MAC payload of an IEEE 802.15.4 frame equals a 6LoWPAN frame, which includes various 6LoWPAN compressed headers and a 6LoWPAN payload. The 6LoWPAN adaptation layer framework (RFC6282) defines LoWPAN IP Header Compression (IPHC) for compressing IPv6 headers, and

LoWPAN Next Header Compression (NHC) for compressing UDP headers and IPv6 Extension Headers. LoWPAN IPHC relies on information pertaining to the entire 6LoWPAN to compress the IPv6 header down to 2 or 3 B. The potential in-line fields directly follow the IPHC and comprise a Hop Limit, a Traffic Class & Flow Label, and a short or extended IPv6 source and destination address. The compressed IPv6 header is followed by a LoWPAN NHC compressed UDP header. A NHC byte indicates the size and presence of potential in-line fields, including a source and destination UDP port number and an UDP checksum. The 6LoWPAN adaptation layer format was also extended by a compression format for RPL headers in RFC8183, called 6LoWPAN Routing Header (6LoRH). It provides a compressed form for the IPv6 RPL Option for Carrying RPL Information in Data-Plane Datagrams (RFC6553) and IPv6 Header for Source Routes (RFC6554). A 6LoRH is expressed as a Type-Length-Value (TLV) field, made up of a Dispatch Value bit pattern, a 6LoRH Type field and a variable length Value field. The RPL Packet Information (RPI)-6LoRH replaces RFC6553 and must be present in every RPL frame. It contains a Sender Rank and an optional RPL Instance ID in case multiple RPL instances are defined. The Source Routing Header (SRH)-6LoRH is a compressed form of the SRH3 defined in RFC6554 and is an optional header that holds the addresses of the remaining hops in the path towards the destination. The SRH-6LoRH is only used in RPL non-storing mode.

The bottom right of Fig. 3 depicts the CoAP frame format, which equals the 6LoWPAN payload. Each CoAP frame comprises a 4 B header including Version, Type, Token Length, Code and Message ID. The header is followed by the Token field and potentially by several CoAP options using a TLV format. Finally, the CoAP payload is separated from the headers by a 1 B Payload Marker. CoAP defines four message types, indicated by the Type field: Confirmable, Non-confirmable, Acknowledgement, and Reset. A Confirmable message requires an Acknowledgement from the receiver or a Reset if it is unable to process the Confirmable message. A Non-confirmable message does not require an Acknowledgement, but the receiver may send a Reset.

5.2. 6LoWPAN Fragmenting

IEEE 802.15.4 defines a MTU for the PSDU, which is either 127 B or 2047 B depending on the chosen PHY. In case of a 127 B MTU, the 6LoWPAN adaptation layer describes the possibility to divide the payload over multiple fragments if the payload does not fit into a single frame. The bottom left of

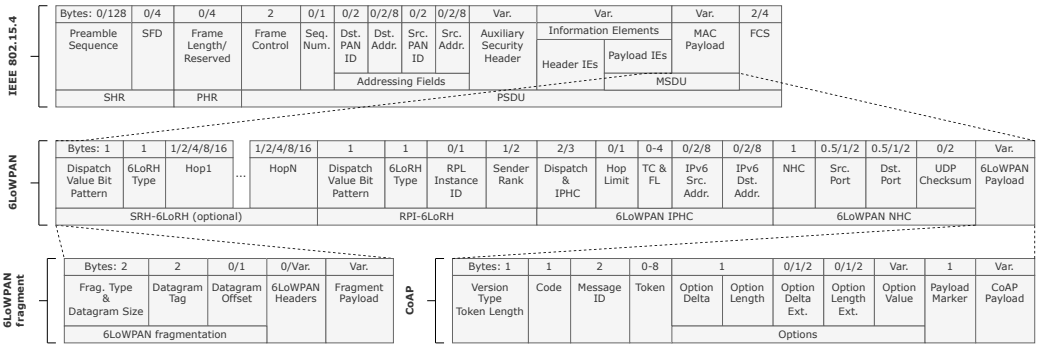


Figure 3: 6TiSCH frame format, comprising of the IEEE 802.15.4 PHY and MAC header (top), 6LoWPAN fixed and optional headers (middle), an optional 6LoWPAN fragmenting header (bottom left), and a CoAP header (bottom right).

Fig. 3 shows the format of a 6LoWPAN fragment and must be seen as either an extension or replacement of the 6LoWPAN frame, as the 6LoWPAN headers are only included in the first fragment but elided in all subsequent fragments. In addition, the Datagram Offset field is elided in the first fragment, resulting in a fragmentation header size of 4 B for the first fragment and 5 B for subsequent fragments. Therefore, next to the 6LoWPAN header, there is only a one-byte difference between the initial and subsequent fragments. Since the Datagram Offset is only able to express multiples of 8 B, all fragments except for the last one must be multiples of 8 B in length.

Knowing this, we are able to derive an equation that calculates the number of fragments and total size of those fragments, given a 6LoWPAN payload size. Eq. (1) calculates the available payload in every fragment, where S_{fpl1} represents the available fragment payload in the first fragment, S_{fplX} the available fragment payload in subsequent fragments, S_{fhdr1} the first fragment header size, S_{fhdrX} the subsequent fragments header size, S_{6mtu} the PSDU MTU, S_{6hdr} the size of 6LoWPAN headers, and S_{mhdr} the size of MAC headers and footers. Since the Datagram Offset only allows multiples of 8 B, the fragment payload size is rounded to 8.

$$\begin{aligned} S_{fpl1} &= \lfloor S_{6mtu} - S_{fhdr1} - S_{6hdr} - S_{mhdr} \rfloor_8 \\ S_{fplX} &= \lfloor S_{6mtu} - S_{fhdrX} - S_{mhdr} \rfloor_8 \end{aligned} \tag{1}$$

Using (1), (2) calculates the total number of fragments N_f, given a 6LoWPAN payload size S_{6pl}. If S_{6pl} fits within S_{6mtu}, the frame can be sent without fragmentation. Else, the number of fragments is dependent on the available

fragment payload for the first (S_{fpl1}) and subsequent (S_{fplX}) fragments.

$$\boldsymbol{N_f}(S_{6pl}) = \begin{cases} 1, & S_{6pl} + S_{6hdr} + \\ & S_{mhdr} \leq S_{6mtu} \\ 1 + \left\lceil \frac{S_{6pl} - S_{fpl1}}{S_{fplX}} \right\rceil, & \text{else} \end{cases} \tag{2}$$

The total size of a frame S_f with a 6LoWPAN payload size S_{6pl} is calculated in (3). As 6LoWPAN fragmenting is considered by taking (2) into account, (3) is valid for fragmented and unfragmented frames. The PHY header's S_{phdr} and MAC header's and footer's size S_{mhdr} are included in every frame or fragment, the 6LoWPAN header's S_{6hdr} size in every frame or in the first fragment, S_{fhdr1} in the first fragment, and S_{fhdrX} in all fragments but the first.

$$\begin{aligned} \boldsymbol{S_f}(S_{6pl}) = {} & S_{6pl} + (S_{mhdr} + S_{phdr}) * \boldsymbol{N_f}(S_{6pl}) \\ & + S_{6hdr} + S_{fhdr1} * \min(\boldsymbol{N_f}(S_{6pl}) - 1, 1) \\ & + S_{fhdrX} * (\boldsymbol{N_f}(S_{6pl}) - 1) \end{aligned} \tag{3}$$

5.3. CoAP Block-Wise Transfer

In case large payloads have to be transmitted, CoAP Block-Wise Transfer (RFC7959) offers the possibility of transmitting multiple blocks of information using multiple request-response pairs. To specify the block-wise transfer characteristics, two CoAP options can be included in the CoAP header: Block1 Option to describe the request payload block-wise transfer and Block2 Option to describe the response payload block-wise transfer. The options can be used in two ways: in descriptive usage, where Block1 is used in a request and Block2 in a response, or in control usage, where Block2 is used in a request (to describe the response block-wise transfer) and Block1 in a response (describing the request block-wise transfer). For this paper, we assume descriptive and control usage, resulting in a CoAP block option in both requests and responses. Although RFC7959 specifies that both messages can be transmitted with CoAP block-wise transfer (using Block 1 and Block 2 at the same time), this is out of scope for this paper. This is in line with *A 3*, where we assumed the bulk of the information flows from a source to a destination.

The number of CoAP blocks N_b, given a CoAP payload S_{cpl} and block size S_{cbs},is calculated in (4), where we assume the source and destination

agree on the block size.

$$\boldsymbol{N_b}(S_{cpl}) = \begin{cases} 1, & S_{cpl} \leq S_{cbs} \\ \left\lceil \frac{S_{cpl}}{S_{cbs}} \right\rceil, & S_{cpl} > S_{cbs} \end{cases} \tag{4}$$

The 6LoWPAN payload size of a single CoAP (block) message is calculated in (5), where S_{chdr} represents the size of the CoAP header (including options and payload marker), S_{copt} the size of a CoAP block option[1], and S_{bpl} the payload of the CoAP block message. If block-wise transfer is not needed, the CoAP block options are excluded and S_{bpl} equals S_{cpl}.

$$\begin{aligned} \boldsymbol{S_{6pl}}(S_{bpl}, S_{cpl}) = S_{chdr} + S_{bpl} \\ + \min(\boldsymbol{N_b}(S_{cpl}) - 1, 1) * S_{copt} \end{aligned} \tag{5}$$

We are now able to calculate S_s, the total number of bytes transmitted by the source in (6). Note that S_{cspl} represents the CoAP payload, transmitted by the source. Since this payload may differ from an optional destination payload S_{cdpl}, we distinguish between S_{cspl} and S_{cdpl} in the following equations. As CoAP block-wise transfer is taken into account by using (4) and (5), (6) is valid for both single CoAP messages and multiple CoAP blocks. If block-wise transfer is needed, all blocks have a payload of S_{cbs} (first term in (6)) except for the last block, which contains the remaining payload (second term in (6)). Otherwise, the CoAP message has a payload of S_{cspl}. Note that by using (3), we also account for 6LoWPAN fragmenting. In general, by correctly choosing the value of S_{cbs}, 6LoWPAN fragmenting should not be necessary. However, since an optional and variable SRH-6LoRH header might be included in the frame (Section 6), or due to a poorly chosen S_{cbs}, CoAP block-wise transfer and 6LoWPAN fragmenting might be needed for the same frame.

$$\begin{aligned} \boldsymbol{S_s}(S_{cspl}) = \boldsymbol{S_f}(\boldsymbol{S_{6pl}}(S_{cbs}, S_{cspl})) * (\boldsymbol{N_b}(S_{cspl}) - 1) \\ + \boldsymbol{S_f}(\boldsymbol{S_{6pl}}(S_{cspl} - (\boldsymbol{N_b}(S_{cspl}) - 1) * S_{cbs}, S_{cspl})) \end{aligned} \tag{6}$$

The total number of bytes transmitted by the destination S_d, given a source CoAP payload S_{cspl} and destination payload S_{cdpl}, is calculated in (7). The

[1]The size of the block option depends on the block number, resulting in a variable S_{copt}. However, for simplicity, this is assumed implicitly.

destination transmits as many CoAP messages as the source transmits CoAP blocks and includes a CoAP block option when needing CoAP block-wise transfer. By using (3), 6LoWPAN fragmenting is also accounted for. For non-confirmable CoAP, the destination has no payload to send, resulting in S_{cdpl} being zero. Nonetheless, the destination still transmits CoAP ACKs because of the CoAP block option's control usage.

$$\boldsymbol{S_d}(S_{cspl}, S_{cdpl}) = \boldsymbol{N_b}(S_{cspl}) * \boldsymbol{S_f}(\boldsymbol{S_{6pl}}(S_{cdpl}, S_{cspl})) \tag{7}$$

6. Multi-Hop Overhead

In multi-hop RPL networks, frames may need to be forwarded along multiple hops to reach their destination. As a consequence, this introduces overhead for all intermediate nodes along the path from source to destination. This multi-hop overhead sub-model takes the unidirectional or bidirectional CoAP payload by the source and destination into account, and calculates the number of transmitted and received bytes for every node along the path from source to destination. Depending on the RPL MOP, the RPL root must be included in the path (non-storing mode) or may be excluded (storing mode), as explained in section 2. To cover all possible scenarios, we assume a Multi-Point to Multi-Point (MP2MP) scenario, where each node is able to reach all other nodes. Fig. 4 shows the trajectory of a CoAP GET-ACK pair along an example three-hop path, including the associated layer-2 Enhanced ACKnowledgements (EACKs)s for each frame. Note that other confirmable CoAP traffic is also possible using the same equations, as is non-confirmable traffic by setting the destination payload S_{dcpl} to zero, but we assume a confirmable CoAP GET-ACK pair in the remainder of this section. As RPL storing mode produces the most general path, we will first consider storing mode, before calculating the overhead in case of non-storing mode. Finally, we take packet loss and associated re-transmissions into account by using the Expected Transmission count (ETX) link metric, used in RPL OFs.

6.1. RPL Storing Mode

Based on Fig. 4, (8) calculates the total number of bytes transmitted by node i ($S_{TX,i}$) as a result of a CoAP GET with payload S_{cdpl} from the destination towards the source and a CoAP ACK with payload S_{cspl} from the source towards the destination. The number of hops between the source and destination is represented by h, $S_{cg,i}$ represents the number of bytes due

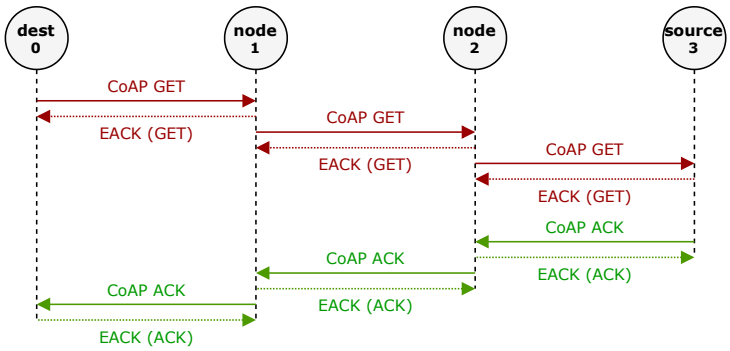


Figure 4: Example multi-hop path of a CoAP GET-ACK pair, including layer-2 EACKs for each frame.

to the forwarding of the CoAP GET message, $S_{eg,i}$ the bytes associated with the EACKs to this message, $S_{ca,i}$ the number of bytes due to the forwarding of the CoAP ACK message, and $S_{ea,i}$ the bytes associated with the EACKs to the CoAP ACK.

$$S_{TX,i} = \begin{cases} S_{cg,i} + S_{ea,i}, & i = 0 \\ S_{cg,i} + S_{ea,i} + S_{ca,i} + S_{eg,i}, & 0 < i < h \\ S_{ca,i} + S_{ea,i}, & i = h \end{cases} \tag{8}$$

Likewise, the total number of received bytes $S_{RX,i}$ by node i are given in (9).

$$S_{RX,i} = \begin{cases} S_{ca,i+1} + S_{eg,i+1}, & i = 0 \\ S_{cg,i-1} + S_{ea,i-1} + S_{ca,i+1} + S_{eg,i+1}, & 0 < i < h \\ S_{ca,i-1} + S_{ea,i-1}, & i = h \end{cases} \tag{9}$$

Before calculating the different terms of (8) and (9), we should take another effect into account that is caused by multi-hop routing: the size of the 6LoWPAN headers is variable depending on the position of the frame along the path. For RPL storing mode, the variable fields are present in the 6LoWPAN IPHC header, namely Hop Limit, IPv6 Source, and Destination address. Eq. (10) calculates the total 6LoWPAN header for RPL storing mode ($S_{6hdr,s}$), where S_{nhc} is the 6LoWPAN NHC size, S_{rpi} the RPI-6LoRH size, S_{iphc} the size of the 6LoWPAN Dispatch and IPHC field, S_{6hl} the size of the Hop Limit, S_{6sa} the size of the IPv6 source address, and S_{6da} the size of the IPv6 destination address. In the first hop, only the IPv6 destination address is included as the IPv6 source address can be obtained from the IEEE

802.15.4 addressing fields. Likewise, the IPv6 destination address is elided in the final hop. Additionally, the IPv6 hop limit is elided in the first hop.

$$\begin{aligned} \boldsymbol{S_{6hdr,s}}(i,h) &= S_{nhc} + S_{rpi} + S_{iphc} \\ &+ \begin{cases} S_{6da}, & i = 0 \\ S_{6hl} + S_{6sa} + S_{6da}, & 0 < i < h-1 \\ S_{6hl} + S_{6sa}, & i = h-1 \end{cases} \end{aligned} \tag{10}$$

By replacing S_{6hdr} in (1)-(3) by (10), eqs. (2), (3), (6), and (7) also become dependent on i and h. This allows us to calculate $S_{cg,i}$ in (11) by using (7) and $S_{ca,i}$ in (12) by using (6). Since the CoAP ACK message travels in the opposite direction of the CoAP GET message, $h - i$ is used instead of i.

$$S_{cg,i} = \boldsymbol{S_d}(S_{cspl}, S_{cdpl}, i, h) \tag{11}$$

$$S_{ca,i} = \boldsymbol{S_s}(S_{cspl}, h-i, h) \tag{12}$$

To calculate the number of bytes due to EACKs, we first calculate the number of EACKs by multiplying the number of fragments with the number of CoAP blocks for each frame. Eqs. (13) and (14) calculate the number of EACKs related to $S_{eg,i}$ and $S_{ea,i}$ respectively and follow a similar approach as (6) and (7). However, instead of using (3) to calculate the size of a 6LoWPAN frame, (2) is used to calculate the total number of fragments, where i and h are also taken into account.

$$N_{eg,i} = \boldsymbol{N_b}(S_{scpl}) * \boldsymbol{N_f}(\boldsymbol{S_{6pl}}(S_{cdpl}, S_{cspl}), i, h) \tag{13}$$

$$\begin{aligned} N_{ea,i} = {} & \boldsymbol{N_f}(\boldsymbol{S_{6pl}}(S_{cbs}, S_{cspl}), h-i, h) * (\boldsymbol{N_b}(S_{cspl}) - 1) \\ & + \boldsymbol{N_f}(\boldsymbol{S_{6pl}}(S_{cspl} - (\boldsymbol{N_b}(S_{cspl}) - 1) * S_{cbs}, S_{cspl}), h-i, h) \end{aligned} \tag{14}$$

To calculate $S_{eg,i}$ and $S_{ea,i}$, it suffices to multiply (13) and (14) with the length of an EACK frame S_{eack}, of which the format is described in Section 7.

6.2. RPL non-storing mode

The above equations are also valid for RPL non-storing mode, except for the addition of a SRH-6LoRH as soon as the frame passes the RPL root. The size of this SRH-6LoRH S_{srh} is calculated in (15), where S_{6srha} is the size of an IPv6 address inside a SRH-6LoRH (defined by the 6LoRH Type

field), and 2 is the size of the Dispatch Value Bit Pattern and 6LoRH Type fields. Note that the number of IPv6 addresses is the remaining number of hops $h - i$ minus one, as the IPv6 address of the final hop is already present as the IPv6 destination address field in the 6LoWPAN IPHC header.

$$\boldsymbol{S_{6srh}}(i, h) = 2 + (h - i - 1) * S_{6srha} \tag{15}$$

This allows us to calculate the total 6LoWPAN header size in case of RPL non-storing mode ($S_{6hdr,ns}$) in (16), where r represents the index of the RPL root along the path. S_{6srh} is only included in all after the root, except for the last.

$$\boldsymbol{S_{6hdr,ns}}(i, h, r) = \boldsymbol{S_{6hdr,s}}(i, h) + \begin{cases} \boldsymbol{S_{6srh}}(i, h), & r \leq i < h - 1 \\ 0, & else \end{cases} \tag{16}$$

Using (16) instead of (10) in (11)-(14) enables us to calculate the total number of transmitted and received bytes by node i in case of non-storing mode, where these equations now also depend on the position of the root along the path r.

6.3. Packet Loss and Re-Transmissions

As packet loss is inherent to LLNs, our model takes the associated retransmissions into account by using the ETX link metric, used in RPL OFs. While other link metrics can be used in OFs, we assume this link metric is available at each node for its preferred parent. The ETX metric is defined in RFC 6551 and provides a discrete value representing the number of expected transmissions of a node to a destination, which is its preferred parent in this case. As the ETX is available locally on every node, it requires to be collected by the CU using INT. However, as only the ETX to a preferred parent of a node is available, we make the assumption each link is symmetric. That way, the same ETX can be used as link metric from preferred parent to the child. If the ETX is included in the model, $S_{cg,i}$ and $S_{ca,i}$ in (8) must be replaced by (17) and (18), where $ETX(i, i+1)$ represent the ETX metric from node i to node $i+1$. As such, the packet is estimated to be transmitted $ETX(i, i+1)$ times, after which the receiver correctly receives it and transmits an EACK. Therefore, (9) and the number of EACKs do not require any modifications.

$$S'_{cg,i} = S_{cg,i} * ETX(i, i+1) \tag{17}$$

$$S'_{ca,i} = S_{ca,i} * ETX(i-1,i) \tag{18}$$

While the above equations only take single-hop re-transmissions into account, it may occur that the number of required re-transmissions exceeds the maximum number of allowed MAC re-transmissions, after which the packet is dropped and higher layers are notified and a multi-hop re-transmission is required. In that case, the accuracy of the model will drop momentarily. However, we chose to not include multi-hop re-transmissions for two reasons. First, if the model wrongfully assumes a multi-hop re-transmissions is required, this will cause an equal drop in the accuracy. Second, if the packet is dropped by the MAC layer, this will probably result in a topology change anyway, after which the accuracy of the model will inevitably drop. However, including multi-hop re-transmissions would only require a minor change to the model, by including the number of allowed MAC re-transmissions.

7. Control Traffic Overhead

This section describes the control traffic overhead sub-model, modelling the TSCH, RPL, and ICMPv6 control frames. The sub-model takes a time interval, during which the number of transmitted and received control frame bytes need to be estimated, as input. In addition, several timing offsets are needed as input, which are gathered in real-time by the CU using the INT, described in Section 8.

7.1. TSCH Control Traffic: EACK and EB

We consider two types of TSCH control messages: EACKs and EBs. For valid frames that are not broadcast, a receiver will reply with an EACK indicating successful reception of the frame. As opposed to a regular ACK, EACKs can carry data and can be secured. Additional content, encapsulated in IEs, can be included in the EACK, such as timing correction in the Timing Correction IE. The frame format of an EACK is identical to the IEEE 802.15.4 frame format depicted in the top of Fig 3. PAN ID compression, Sequence Number compression and Security fields are set to the same value as the corresponding fields of the acknowledged frame.

Nodes use EBs to enable other nodes to join the network and send periodic updates of TSCH parameters, such as scheduling and synchronization information. Once connected to the network, each node generates an EB with a time period T_{eb} and sends this EB during the next advertisement

slot, dedicated for EB transmission. The EB frame format is identical to the IEEE 802.15.4 frame depicted on the top of Fig. 3. EBs must include the address of the transmitting device in the Source Address field and may use IEs to inform receiving nodes with the required information. Although a period at which EBs are generated (T_{eb}) needs to be defined by the protocol, the actual period in between EB transmissions may vary depending on the TSCH schedule. After all, EBs can only be transmitted during TSCH advertisement slots. Therefore, the number of EB transmissions that occur within a certain time interval T_i depends on the relative positioning of T_i to the advertisement slots and EB generation times. This is illustrated in Fig. 5, where each node is assigned a single advertisement slot per slotframe with duration T_{sf}. Idle advertisement slots are highlighted in green, active advertisement slots in red, and EB generation times by red arrows. In this simple example, although two EBs are generated during T_i, only a single EB is transmitted. We calculate the number of transmitted EBs within a time interval T_i in (19), taking the relative positioning of T_i to the EB generation times and advertisement slots into account. As illustrated in Fig. 5, T_{o1} and T_{o2} represent the time offsets between the start of T_i and the previous EB generation and advertisement slot respectively, whereas $T_{o1'}$ and $T_{o2'}$ are similar time offsets between the end of T_i and the last EB generation and advertisement slot within T_i. The first term in (19) calculates the total number of generated EBs within T_i, taking the time offset T_{o1} into account. The final term accounts for the positioning of the advertisement slots at the beginning and end of T_i: if $T_{o2} > T_{o1}$, and extra EB transmission occurs, whereas if $T_{o2'} > T_{o1'}$, the final generated EB will not be transmitted within T_i resulting in one less EB transmission. Note that (19) only depends on T_{o1} and T_{o2}, since $T_{o1'}$ and $T_{o2'}$ can be calculated using (20), where T equals T_{eb} for T_{o1} and T_{sf} for T_{o2}. T_{o1} and T_{o2} must be obtained once for each node using INT, described in Section 8.

$$\boldsymbol{N_{eb}}(T_i, T_{o1}, T_{o2}) = \left\lfloor \frac{T_i - (T_{eb} - T_{o1})}{T_{eb}} \right\rfloor + \frac{\text{sign}(T_{o2} - T_{o1}) + \text{sign}(T_{o2'} - T_{o1'})}{2} \tag{19}$$

$$T_{o'} = \left\lfloor \frac{T_i - (T - T_o)}{T} \right\rfloor * T - T_o - T_i \tag{20}$$

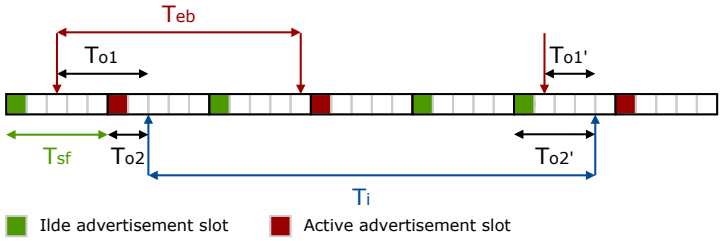


Figure 5: Periodical EB transmission for a defined EB period T_{EB} and a slotframe size T_{sf} during a time interval T_i. The number of transmitted EBs (active advertisement slots) may vary to the number of generated EBs (red arrows) during T_i, depending on the relative positioning of the advertisement slots and EB generation times to T_i.

While (19) is suited for small intervals, when using large values of T_i, the time offsets become negligible and (19) can be replaced by (21).

$$\boldsymbol{N_{eb}}(T_i) = \frac{T_i}{T_{eb}} \tag{21}$$

The total number of transmitted bytes due to EBs can therefore be calculated in (22), where S_{eb} represents the size of an EB. To calculate the total number of received bytes due to EBs, it suffices to use (22) for every neighbour the node listens to, which must be obtained using INT.

$$\boldsymbol{S_{TX,eb}}(T_i, T_{o1}, T_{o2}) = \boldsymbol{N_{eb}}(T_i, T_{o1}, T_{o2}) * S_{eb} \tag{22}$$

7.2. RPL Control Traffic: DIO and DAO

RPL employs five types of control messages: a DODAG Information Solicitation (DIS) message, DODAG Information Object (DIO) message, Destination Advertisement Object (DAO) message, DAO ACKnowledgement (DAO-ACK) message, and Consistency Check (CC) message. As we assume a deployed and converged network (*A 2*), DIS and CC messages are not considered in our model because they are used during network formation. Fig. 6 shows the frame format for a DIO (left), DAO (middle), and DAO-ACK (right) message. As RPL control messages are ICMPv6 messages, each message type includes an ICMPv6 header and a message body, comprising a message base and a variable number of options. The frame formats of Fig. 6 must be seen as an addition to the 6LoWPAN frame format depicted in the middle of Fig. 3, where the ICMPv6 header is encoded using NHC. Depending on RPL configuration parameters, multiple options can be

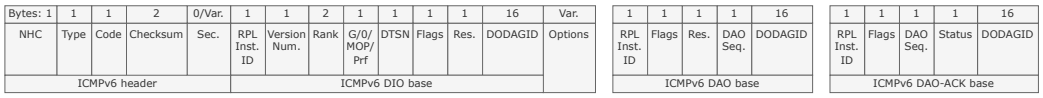


Figure 6: ICMPv6 RPL control messages: DIO (left), DAO (middle), and DAO-ACK (right). DAOs and DAO ACKs also contain the ICMPv6 header and Options, shown in the DIO message.

included in the control messages. For more information on these options, the reader is referred to RFC6550.

In a stable network, nodes use DIOs to maintain the DODAG and upward routes, and distribute configuration parameters and related metrics. They multicast DIOs to all RPL nodes using a Trickle timer (RFC6206). As the network is fully converged (*A 2*), nodes use the maximum interval I_{max}. The Trickle algorithm states that nodes choose a random moment within $[I_{max}/2, I_{max}]$ since the start of the interval to generate a DIO message. Because of this random nature, it is impossible to determine the exact amount of transmitted DIOs within a certain time interval T_i. However, the number of transmitted DIOs can be approximated by (23), where T_{dio} equals $3I_{max}/4$.

$$\boldsymbol{N_{dio}}(T_i) = \frac{T_i}{T_{dio}} \tag{23}$$

The total number of transmitted bytes due to DIOs is calculated in (24), where S_{dio} represents the size of a DIO. It must be noted that we did not take the redundancy constant k into account, which suppresses a DIO transmission if k DIOs where received during the current Trickle interval. However, the effect of k has already been modelled in [16]. To calculate the number of received bytes due to DIOs, (24) must be used for every neigbour the node listens to, obtained using INT.

$$\boldsymbol{S_{dio}}(T_i) = \boldsymbol{N_{dio}}(T_i) * S_{dio} \tag{24}$$

Contrary to DIOs, DAOs are used for downward route discovery and maintenance in both storing and non-storing mode. Every device must only generate a new DAO before the expiration of the Path Lifetime, of which the duration is implementation specific. Therefore, the interval in between DAOs might be random as well, similar to DIOs. Nonetheless, as a DAO generation results in more transmitted and received bytes than a DIO generation (due to DAO-ACKs and forwarding), a time offset to the last transmitted DAO

can increase the precision of the model, especially for short time intervals. In non-storing mode, DAOs are unicast to the root to construct SRHs. The number of DAO transmissions during a time interval T_i in non-storing mode ($N_{dao,ns}$) is therefore given in (25), where T_o represents the offset to the previous DAO transmission and the beginning of T_i, and T_{dao} is equal to the (average) interval in between DAOs.

$$\boldsymbol{N_{dao,ns}}(T_i, T_o) = \left\lfloor \frac{T_i - (T_{dao} - T_o)}{T_{dao}} \right\rfloor \tag{25}$$

Similar to EBs, including T_o in (25) is suited for small intervals. If the time offset to the previously transmitted DAO is not available (e.g. because it is not included in the telemetry), (25) falls back to (26).

$$\boldsymbol{N_{dao,ns}}(T_i) = \frac{T_i}{T_{dao}} \tag{26}$$

Since DAOs are unicast to the RPL root, all devices along the path towards the root transmit and receive DAOs and EACKs. To calculate the number of transmitted and received bytes per node, we make use of (8) and (9), where $S_{cg,i}$ is replaced by $S_{d,i}$, calculated in (27). We use (3) to calculate the size of the frame and replace S_{6hdr} by (16) to also include routing headers dependent on i, h, and r. S_{dao} represents the size of the DAO as depicted in Fig. 6. Since the frame is destined to the RPL root, the position of the root along the path is chosen to be h, resulting in no SRH.

$$S_{d,i} = \boldsymbol{S_f}(S_{dao}, i, h, h) \tag{27}$$

If DAOs need to be confirmed by DAO-ACKs, $S_{ca,i}$ in (8) and (9) is replaced by $S_{da,i}$, given in (28). Instead of using h as the position of the root along the path, we now use 0 which results in a SRH inclusion. S_{daoa} represents the size of the DAO-ACK. In case DAO-ACKs are disabled, it suffices to remove $S_{da.i}$ and the associated EACKs from (8) and (9).

$$S_{da,i} = \boldsymbol{S_f}(S_{daoa}, i, h, 0) \tag{28}$$

Contrary to non-storing mode, devices unicast DAOs to their DAO parents when using storing mode, who will be triggered to send a DAO themselves if the received DAO contains updated information. Expiration of the Path Lifetime automatically results in updated DAO information, because the

Path Sequence will be increased as a result. However, devices should wait for a delay of *DelayDAO* before triggering a DAO in response to an updated DAO, in order to aggregate information from multiple received DAOs. Each device can have a different *DelayDAO* and the correct choice of *DelayDAO* can reduce the number of transmitted DAOs. If *DelayDAO* is non-zero, the number of DAO transmissions per Path Lifetime, the associated timings, and the DAO parents per node can be collected using INT. In that case, the number of transmitted DAOs for storing mode $N_{dao,s}$ is calculated in (29), where N_{dp} indicates the number of DAO parents and D the number of DAO transmissions per Path Lifetime. Note that all of the D DAOs will be sent to each of the N_{dp} DAO parents. $T_{o,j}$ represents the offset from the beginning of T_i to the previous DAO transmission, associated with the j^{th} DAO transmission per Path Lifetime. If *DelayDAO* is zero, D equals the number of DAO parents. If no timing offset(s) is/are collected, (26) should be used inside the sum operator. Collecting timing offsets is therefore optional for both DAO and EB estimation, as alternative equations are available.

$$\boldsymbol{N_{dao,s}}(T_i) = N_{dp} * \sum_{j=0}^{D-1} \boldsymbol{N_{dao,ns}}(T_i, T_{o,j}) \tag{29}$$

As was the case for non-storing mode, (8) and (9) can be used to calculate the total transmitted and received bytes respectively, for a single DAO transmission. We again replace $S_{cp,i}$ by $S_{d,i}$, which is now given by (30), where S_{6hdr} in (3) is calculated using (10). Since DAOs in storing mode are unicast to DAO parents (i.e. single-hop), h is set to 1.

$$S_{d,i} = \boldsymbol{S_f}(S_{dao}, i, 1) \tag{30}$$

Similar to non-storing mode, $S_{ca,i}$ can be replaced by (31) if DAO-ACKs are enabled, or omitted if disabled.

$$S_{da,i} = \boldsymbol{S_f}(S_{daoa}, i, 1) \tag{31}$$

7.3. ICMPv6 Control Traffic: 6LoWPAN ND

6LoWPAN Neighbour Discovery (ND) (RFC6775) provides an alternative to classic IPv6 ND for LoWPANs by limiting the use of multicast, enabling multi-hop distribution of prefix and 6LoWPAN context, and introducing several new ICMPv6 options. Six ICMPv6 messages are defined:

Neighbour Solicitation (NS), Neighbor Advertisement (NA), Router Solicitation (RS), Router Advertisement (RA), Duplicate Address Request (DAR) and Duplicate Address Confirmation (DAC). Messages follow the same frame format as depicted in Fig. 6, without the DIO, DAO, or DAO-ACK base, and can include three possible options: Address Registration Option (ARO), 6LoWPAN Context Option (6CO), and Authoritative Border Router Option (ABRO). Each 6LoWPAN Node (6LN) (hosts and 6LoWPAN Routers (6LRs)) periodically transmits unicast NS messages to each of its 6LRs, stored in its neighbour cache, for re-registration and Neighbor Unreachability Detection (NUD) purposes. The period of transmission is given by the Registration Lifetime, which is defined in the ARO included in a NS. In response to the NS, the receiving 6LR responds with a NA message to the 6LN and updates the corresponding entry of the 6LN in the 6LBR's Duplicate Address Detection (DAD) table, which results in a DAR from 6LR to 6LBR and a DAC vice versa. The number of NS transmissions within a time interval T_i for a single Neighbour Cache Entry (NCE) can be calculated using (25), by replacing T_{dao} with the registration lifetime T_{rl}. The total number of transmitted and received bytes due to NS, NA, DAR, and DAC are calculated using (8) and (9) as all these messages are unicast. Depending on the type of message, source, destination, and RPL MOP, the correct values of $S_{cg,i}$, $S_{ac,i}$, $S_{eg,i}$, and $S_{ea,i}$ need to be calculated, similar to RPL control traffic. These calculations need to be performed for every entry in the neighbour cache, which is collected for every node using INT.

In a route-over topology, RAs are multicast by 6LBRs and 6LRs to other 6LRs, comprising an ABRO, 6CO, and Prefix Information Object (PIO). The RAs should be transmitted with a period sufficiently smaller than ABRO Valid Lifetime such that missing an RA does not result in removing all 6LBR information. RFC6775 specifies substitutable features, which may be substituted by a routing protocol, such as RPL. One of those features is the multihop distribution of prefix and 6LoWPAN header compression, comprised in the ABRO and 6CO. As DIOs provide this functionality, we assume the RA functionality is taken care of by DIOs.

8. In-band Network Telemetry

As discussed in Section 4, the CU is in need of real-time telemetry from each node to provide an accurate estimation of future network traffic. The implementation of INT in [30] only allows for telemetry distribution to the

Table 2: Required (top) and optional (bottom) telemetry per RPL Mode Of Operation (MOP).

Telemetry	**Non-storing MOP**	**Storing MOP**	**Size**
Neighbours	✓	✓	N B
Preferred parent	✗	✓	1 B
Routing table	✗	✓	$2R$ B
Time source	✓	✓	1 B
DAO parents	✗	✓	D B
ETX	✓	✓	2 B
TX ASN	✓	✓	2 B
EB ASN	✓	✓	4 B
DAO ASN	✓	✗	3 B

6LBR in non-storing mode of RPL, because telemetry is added to frames which eventually pass the 6LBR (RPL root) due to the nature of non-storing mode. In storing mode, however, frames do not always pass the 6LBR. We therefore extend the solution in [30] to storing mode by making use of INT in DAOs. Although DAOs are not unicasted to the root, the reception of a DAO by a DAO parent triggers the transmission of a new DAO to its own DAO parent, until a DAO reaches the root. Therefore, two adaptations are needed: (i) nodes are only allowed to insert telemetry in DAOs, and (ii) the telemetry in received DAOs needs to be copied to transmitted DAOs. Inserting telemetry in DAOs has an additional advantage when used in our model: DAOs are generated periodically in a stable network but are also triggered upon a topology change, which allows the model to quickly adapt to a topology change while limiting the telemetry overhead during stable periods. As such, an adaptive INT update interval is achieved.

Since the CU is connected to the 6LBR, some essential information might already be present, depending on the MOP. Additionally, some telemetry is required for correct execution of the model, whereas some telemetry is optional to increase the accuracy of the model. Table 2 lists the required (top) and optional (bottom) telemetry for each MOP, including the telemetry size, where N, R, and D represent the number of neighbours, routes, and DAO parents respectively. Each node is assigned a 1 B ID by the 6LBR (e.g., by using a hash table of the MAC address), which results in 256 possible

nodes in the network. Naturally, the preferred parent and routing table are not required for non-storing mode as the 6LBR already possesses this information. In contrast, nodes in storing mode retain their own routing table so must forward this to the 6LBR. Additionally, DAO parents are only relevant for storing mode.

As discussed in Section 7, timing offsets to the previous EB and DAO transmission/generation can increase the accuracy of the prediction, especially for shorter prediction intervals. In addition, ETX can improve the accuracy of the model in case of packet loss. Inserting these telemetry options is, however, optional. To transmit the timing offsets, the Absolute Slot Number (ASN) is used as it is shared by every node in the network. The TX ASN refers to the moment at which the frame, encapsulating the telemetry, is transmitted. The EB and DAO ASNs are the time offsets (in ASNs) relative to the TX ASN. Note that two EB ASNs need to be included, for T_{o1} and T_{o2} in (19).

9. Evaluation

This section evaluates the accuracy of our model under various conditions, including topology changes and packet loss. Additionally, we examine the effects of optional telemetry, INT update interval, and prediction interval for storing and non-storing modes. To test the model, we simulated a network of ten nodes using the Cooja network simulator of Contiki-NG. Nine client nodes periodically transmit CoAP GET requests to a server, which responds with a CoAP ACK message. The server was chosen differently from the 6LBR to account for MP2MP traffic, and the 6LBR also functions as a client and polls for server updates. Table 3 lists all necessary parameters for the analytical model used in the simulations, from top to bottom: data traffic, control traffic, header configuration, and TSCH network configuration parameters. In terms of data traffic parameters, clients request a 30 B update every 3 min. Control traffic parameters use default values in Contiki-NG. According to the Trickle Algorithm, the actual DIO interval is a random value between 524 s and 1048 s, given a maximum Trickle interval of 1048 s (Section 7). The DAO interval is implementation-specific and results in a random value between 15 min and 22.5 min for a Prefix Lifetime of 30 min, according to the Contiki-NG implementation. In the simulations, each node has a single DAO parent that coincides with its preferred parent. The DelayDAO parameter is set to 0 s to limit INT overhead. Note that 6LoWPAN

ND messages are not considered in the simulations due to the current version of Contiki-NG not implementing 6LoWPAN ND. However, as described in Section 7, this could be easily included in the model using the equations for other control traffic. The network header configuration parameters allow for solving the single-hop and multi-hop submodel in Sections 5 and 6, respectively. As discussed in Section 5.1, the MAC header size depends on the frame type. Therefore, the size of control traffic frames already includes the MAC header size, and the header size for data frames is defined as S_{mhdr} in Table 3. Finally, the bottom part of Table 3 lists TSCH configuration parameters. We use a slot size of 10 ms with the Orchestra Sender-Based scheduling function, where the EB slotframe size is 397, the Broadcast slotframe size is 31, and the Sender-based Unicast slotframe is 17. For further details on the Orchestra Sender-Based scheduling function, refer to [4].

For all simulations, we calculate the average accuracy every second over the 10 simulated nodes. The average accuracy of is calculated using (32), where $P_{tx,i}$ and $P_{rx,i}$ represent the predicted TX and RX bytes of node i, and $A_{tx,i}$ and $A_{rx,i}$ the actual TX and RX bytes.

$$a = \frac{\sum_{i=1}^{10} |P_{tx,i} + P_{rx,i} - A_{tx,i} - A_{rx,i}|}{\sum_{i=1}^{10} (A_{tx,i} + A_{rx,i})} \tag{32}$$

9.1. Effect of Prediction Interval

In our initial simulations, we examine a stable, fully converged network over a duration of six hours. Fig. 7 presents the distribution of the accuracy for prediction intervals of 1 min, 15 min, and 1 h. We display the accuracy for storing and non-storing modes, employing only the default telemetry (Default) and also with optional timing telemetry enabled (Opt. INT), as listed in Table 2. For every simulation, we use an average INT update interval of 15 min for both storing and non-storing modes. The results indicate that the accuracy increases with larger prediction intervals, reaching 99.37 % and 99.00 % for storing and non-storing modes, respectively. This increase is due to the averaging out of timing errors at the beginning and end of the prediction interval by more prediction data. However, even for a small prediction interval of 1 min, the median accuracy remains at 91.46 % and 92.78 % for storing and non-storing modes, respectively. Moreover, when optional timing telemetry is enabled, the median accuracy rises to 95.28 % and 94.88 %, albeit at the cost of introducing more deviant outliers. On the other hand,

Table 3: Parameters of the analytical model, used in the simulations. From top to bottom: data parameters, control traffic parameters, network header parameters, and TSCH network configuration paremeters.

Parameter	**Value**
CoAP GET payload (S_{cdpl})	7 B
CoAP ACK payload (S_{cspl})	30 B
CoAP interval	3 min
EB size (S_{eb})	37 B
EB period (T_{eb})	16 s
DIO size (S_{dio})	96 B
Max. Trickle interval (I_{max})	1048 s
DAO size (S_{dao})	67/85 B
DAO ACK size (S_{daoa})	34 B
Prefix Lifetime (S_{pl})	30 min
DAO parents	1
DelayDAO	0 s
PHY header size (S_{phdr})	5 B
MAC header size (S_{mhdr})	23 B
Dispatch & IPHC bytes (S_{iphc})	2 B
IPv6 address size (S_{6da}, S_{6sa}, S_{6srha})	8 B
6LoWPAN NHC size (S_{nhc})	7 B
CoAP header size (S_{chdr})	5 B
6LoWPAN MTU (S_{6mtu})	127 B
CoAP Block size (S_{cbs})	64 B
TSCH Scheduling Function	Orchestra TSCH-SB-397-31-17
TSCH slot duration (T_{ts})	10 ms

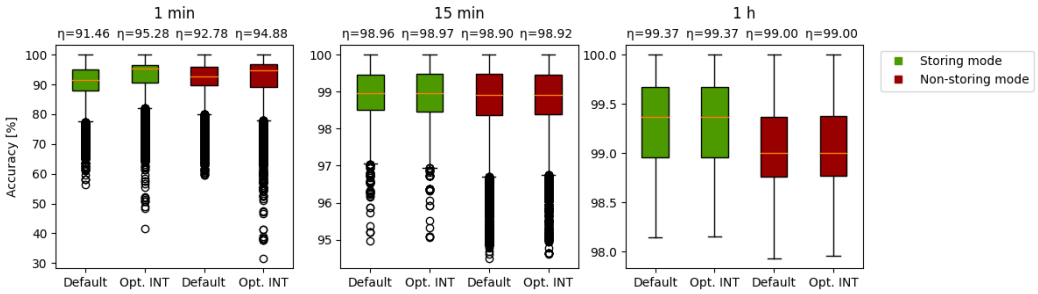


Figure 7: Effect of the prediction interval and optional INT on the accuracy of the model for storing mode (green) and non-storing mode (red). Predictions were calculated every second for six hours in a converged network, for a 1 min (left), 15 min (middle), and 1 h (right) prediction interval. As shown, the accuracy of the model increases for larger prediction interval since timing errors at the edges of the interval are averaged out. Therefore, enabling optional INT for shorter prediction intervals of 1 s increases the accuracy significantly, while having a negligible effect for longer intervals of 15 min and 1 h.

for larger prediction intervals, the optional timing telemetry does not have a significant effect. This is because the optional telemetry is intended to minimize timing-related errors at the start and end of the prediction interval, resulting in substantial advantages for shorter prediction intervals where such errors are more crucial. In conclusion, the accuracy of our model declines for smaller prediction intervals but can be enhanced by enabling optional timing telemetry. For larger intervals, the accuracy improves, and optional timing telemetry does not significantly enhance accuracy.

9.2. Effect of Packet Loss

To evaluate the impact of packet loss on accuracy in stable network operation, we conducted additional simulations with Packet Reception Ratio (PRR) set to 100 %, 90 %, and 80 %. These simulations used a prediction interval and average INT update interval of 15 min. Fig. 8 illustrates the accuracy for storing and non-storing modes and demonstrates the effect of using ETX. As anticipated, enabling ETX has a negligible effect on accuracy when PRR is 100 %. However, it significantly improves accuracy in cases where PRR is less than 100 %. Even with a PRR of 90 % or 80 %, our model achieves a median accuracy of approximately 97 %. Consequently, ETX telemetry should always be employed to enhance accuracy, particularly in LLNs with suboptimal PRRs.

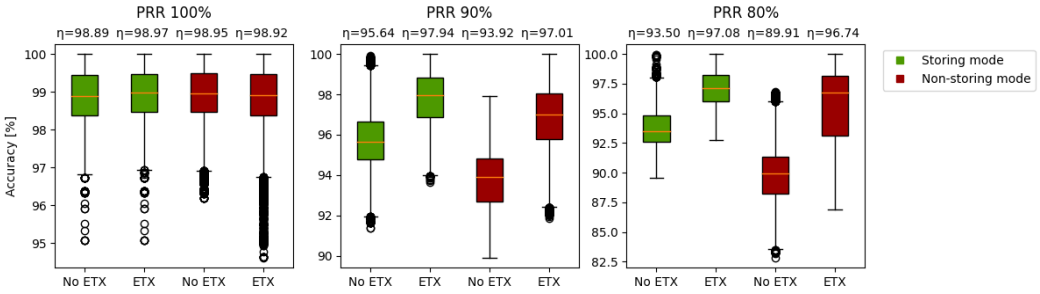


Figure 8: Effect of packet loss and ETX telemetry on the accuracy of the model for storing mode (green) and non-storing mode (red). Predictions were calculated every second for six hours in a stable network for a PRR of 100 % (left), 90 % (middle), and 80 % (right). As shown, enabling ETX telemetry significantly increases the accuracy when packet loss is introduced, obtaining a median accuracy of approximately 97 %.

9.3. Effect of Topology Changes

While the previous simulations assumed a stable network, our model is also capable of reacting to a topology change. To demonstrate this, we introduced a topology change in a fully converged network by moving three nodes to the other side of the network, which required them to rejoin with different preferred parents and neighbours. Predictions were calculated every second for two hours since the start of the topology change, with a prediction interval of 15 min. The distribution of the accuracy during this time is depicted in Fig. 9a for storing mode and in Fig. 9b for non-storing mode. During the topology change, the model's accuracy initially dropped but gradually increased again as the CU received the updated topology through INT. This explains why there are many outliers in the accuracy distribution, as they mostly represent the accuracy during the topology change. We considered different INT update intervals to determine their impact on the model's responsiveness. The blue line on the right vertical axis indicates the INT overhead, which is the number of transmitted INT bytes per node per hour for the given INT update interval.

We used DAOs as telemetry carriers in the storing mode simulations and data traffic in non-storing mode simulations to demonstrate the difference between them for piggybacking INT to the 6LBR. In storing mode, we considered average INT update intervals of 60, 30, and 15 min. Due to a random DAO interval between 15 and 22.5 min, the lowest achievable update interval was 15 min. We also considered an adaptive update interval, where each triggered DAO piggybacks INT. As shown in Fig. 9a, increasing the INT

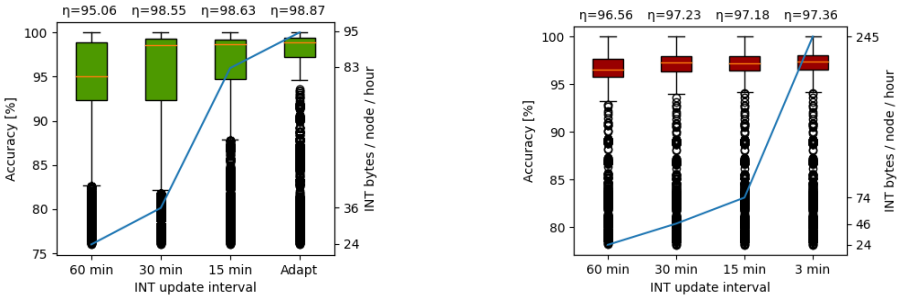
update interval improved the prediction accuracy significantly, but it also increased the INT overhead. However, with an adaptive update rate, the accuracy improved notably with only a limited increase in INT overhead. This is because, during a topology change, more DAOs will be triggered because of downward route changes, causing the update interval to momentarily increase before converging back to 15 min during stable operation.

In non-storing mode, the effect of the INT update interval was very different from storing mode, as shown in Fig. Fig. 9b). We used the same update intervals as in storing mode, in addition to an interval of 3 min, which is the lowest achievable interval equal to the CoAP interval. The INT update rate had limited to no impact on the prediction accuracy, which can be explained by the fact that in non-storing mode, the 6LBR is already aware of the network topology, even without INT. Only the neighbours, time source, and optional telemetry need to be collected (Table 2).

In conclusion, our model is capable of reacting to changes in network topology. For storing mode, choosing an adaptive INT update interval can increase the prediction accuracy with only a limited increase in INT overhead. However, the INT update rate has limited impact on the prediction accuracy for non-storing mode since the CU is already aware of the topology change without INT.

10. Conclusion

The ability to gather insights about a network is critical for a central network manager to detect and resolve potential issues such as congestion, duty cycle regulation breaches, and battery depletion. In this paper, we propose a model that can calculate the network traffic overhead in a 6TiSCH network, using in-band network telemetry to predict the number of transmitted and received bytes. Our model covers the entire 6TiSCH stack, including TSCH, RPL, 6LoWPAN, and CoAP, and calculates the introduced overhead in terms of control traffic, single-hop, and multi-hop communication. Through simulations, we demonstrate that our model provides accurate predictions under various circumstances, such as sudden topology changes and packet loss. During topology changes, non-storing mode predictions retain high accuracy regardless of the telemetry update interval, while an adaptive update interval significantly improves prediction accuracy in storing mode while minimizing overhead. In a converged network, accuracy increases for larger prediction intervals, and optional timing telemetry can be used to



(a) Storing mode

(b) Non-storing mode

Figure 9: Effect of a topology change for (a) storing and (b) non-storing mode on the accuracy of the model for different INT update intervals. Boxplots show the accuracy of the model for different update intervals with the associated number of INT bytes per node per hour indicated by blue lines. Predictions are calculated every second for two hours since the start of the topology change, with a prediction interval of 15 min. In storing mode (a), INT is collected by DAOs, while in non-storing mode (b), data traffic is used to piggyback telemetry. As shown, increasing the INT update interval significantly improves the accuracy in storing mode, while having a negligible effect in non-storing mode since the 6LBR knows the network topology without requiring INT.

improve accuracy for shorter intervals. We showed that in-band network telemetry is essential to construct a topology at the CU, to restrict computation overhead on the nodes while minimizing communication overhead. Our open-source model can be used to prevent network outages due to congestion, duty-cycle regulation breaches, battery depletion, etc. Furthermore, it can be easily extended with a machine learning model to predict traffic generation patterns or with energy measurements for different TSCH slot types to predict node energy consumption precisely.

Acknowledgment

Part of this research was funded by the Flemish FWO SBO S001521N IoBaLeT (Sustainable Internet of Batteryless Things) project.

References

[1] M. L. N. Shilpa P. Khedkar, R. Aroul Canessane, Prediction of Traffic Generated by IoT Devices Using Statistical Learning Time Series Algorithms, Wireless Communications and Mobile Computing (2021) 12doi:10.1155/2021/5366222.

[2] Dries Van Leemput, 6tisch_traffic_model.
URL `https://github.com/imec-idlab/6tisch_traffic_model.git`

[3] IEEE Standard for Local and Metropolitan Area Networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC Sublayer, Ieee standard 802.15.4-2020 (July 2020).

[4] S. Duquennoy, B. Al Nahas, O. Landsiedel, T. Watteyne, Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH, in: Proceedings of the ACM Conference on Embedded Networked Sensor Systems, 13th Edition, 2015, p. 337–350. doi:10.1145/2809695.2809714.

[5] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang, K. S. J. Pister, A Realistic Energy Consumption Model for TSCH Networks, IEEE Sensors Journal 14 (2) (2014) 482–489. doi:10.1109/JSEN.2013.2285411.

[6] G. Daneels, E. Municio, B. Van de Velde, G. Ergeerts, M. Weyn, S. Latré, J. Famaey, Accurate Energy Consumption Modeling of IEEE 802.15.4e TSCH Using Dual-Band OpenMote Hardware, Sensors 18 (2) (2018). doi:10.3390/s18020437.

[7] C. Ouanteur, L. Bouallouche-Medjkoune, D. Aïssani, An Enhanced Analytical Model and Performance Evaluation of the IEEE 802.15.4e TSCH CA, Wireless Personal Communications 96 (09 2017). doi:10.1007/s11277-017-4241-0.

[8] H. Hajizadeh, M. Nabi, R. Tavakoli, K. Goossens, A Scalable and Fast Model for Performance Analysis of IEEE 802.15.4 TSCH Networks, in: 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2019, pp. 1–7. doi:10.1109/PIMRC.2019.8904420.

[9] S. Scanzio, M. G. Vakili, G. Cena, C. G. Demartini, B. Montrucchio, A. Valenzano, C. Zunino, Wireless Sensor Networks and TSCH: A Compromise Between Reliability, Power Consumption, and Latency, IEEE Access 8 (2020) 167042–167058. doi:10.1109/ACCESS.2020.3022434.

[10] D. De Guglielmo, B. Al Nahas, S. Duquennoy, T. Voigt, G. Anastasi, Analysis and Experimental Evaluation of IEEE 802.15.4e TSCH CSMA-

CA Algorithm, IEEE Transactions on Vehicular Technology 66 (2) (2017) 1573–1588. doi:10.1109/TVT.2016.2553176.

[11] S. B. Yaala, F. Théoleyre, R. Bouallegue, Performance Modeling of IEEE 802.15.4-TSCH with Shared Access and ON-OFF traffic, in: 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), 2018, pp. 352–357. doi:10.1109/IWCMC.2018.8450358.

[12] D. De Guglielmo, S. Brienza, G. Anastasi, A Model-based Beacon Scheduling algorithm for IEEE 802.15.4e TSCH networks, in: 2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2016, pp. 1–9. doi:10.1109/WoWMoM.2016.7523517.

[13] O. Gaddour, A. Koubâa, S. Chaudhry, M. Tezeghdanti, R. Chaari, M. Abid, Simulation and performance evaluation of DAG construction with RPL, in: Third International Conference on Communications and Networking, 2012, pp. 1–8. doi:10.1109/ComNet.2012.6217747.

[14] H. R. Kermajani, C. Gomez, Modeling the network convergence time in RPL in error-prone, IEEE 802.15.4 chain topology multihop networks, in: 2014 11th International Symposium on Wireless Communications Systems (ISWCS), 2014, pp. 365–369. doi:10.1109/ISWCS.2014.6933379.

[15] M. Becker, K. Kuladinithi, C. Görg, Modelling and Simulating the Trickle Algorithm, in: K. Pentikousis, R. Aguiar, S. Sargento, R. Agüero (Eds.), Mobile Networks and Management, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 135–144.

[16] H. Kermajani, C. Gomez, M. H. Arshad, Modeling the Message Count of the Trickle Algorithm in a Steady-State, Static Wireless Sensor Network, IEEE Communications Letters 16 (12) (2012) 1960–1963. doi:10.1109/LCOMM.2012.111612.121232.

[17] B. Bannour, A. Lapitre, Model Checking of Trickle-based IoT Dissemination, in: 2020 9th Mediterranean Conference on Embedded Computing (MECO), 2020, pp. 1–6. doi:10.1109/MECO49872.2020.9134251.

[18] C. M. G. Algora, E. O. Guerra, S. Montejo-Sánchez, E. M. G. Fernández, K. Steenhaut, A Theoretical Association Time Model for IEEE 802.15.4

TSCH Networks, IEEE Communications Letters 25 (2) (2021) 656–659. doi:10.1109/LCOMM.2020.3032674.

[19] C. Vallati, S. Brienza, G. Anastasi, S. K. Das, Improving Network Formation in 6TiSCH Networks, IEEE Transactions on Mobile Computing 18 (1) (2019) 98–110. doi:10.1109/TMC.2018.2828835.

[20] A. Kalita, M. Khatua, Channel Condition Based Dynamic Beacon Interval for Faster Formation of 6TiSCH Network, IEEE Transactions on Mobile Computing (2020) 1–1doi:10.1109/TMC.2020.2980828.

[21] J. Vera-Pérez, J. Silvestre-Blanes, V. Sempere-Payá, TSCH and RPL Joining Time Model for Industrial Wireless Sensor Networks, Sensors 21 (11) (2021). doi:10.3390/s21113904.

[22] M. Kubaszek, J. Macheta, L. Krzak, C. Worek, The analysis of energy consumption in 6TiSCH network nodes working in sub-GHz band, International Journal of Electronics and Telecommunications vol. 66 (No 1) (2020) 201–210. doi:10.24425/ijet.2020.131864.

[23] D. Hauweele, R.-A. Koutsiamanis, B. Quoitin, G. Z. Papadopoulos, Thorough Performance Evaluation Analysis of the 6TiSCH Minimal Scheduling Function (MSF), Journal of Signal Processing Systems (June 2021). doi:10.1007/s11265-021-01668-w.

[24] A. Weigel, V. Turau, An Analytical Model of 6LoWPAN Route-Over Forwarding Practices, in: S. Guo, J. Lloret, P. Manzoni, S. Ruehrup (Eds.), Ad-hoc, Mobile, and Wireless Networks, Springer International Publishing, Cham, 2014", pp. 279–289.

[25] H. A. Al-Kashoash, F. Hassen, H. Kharrufa, A. H. Kemp, Analytical modelling of congestion for 6LoWPAN networks, ICT Express 4 (4) (2018) 209–215. doi:https://doi.org/10.1016/j.icte.2017.11.001.

[26] R. Herrero, Analytical model of IoT CoAP traffic, Digital Communications and Networks 5 (2) (2019) 63–68. doi:https://doi.org/10.1016/j.dcan.2018.07.001.

[27] A. Ludovici, P. D. Marco, A. Calveras, K. H. Johansson, Analytical Model of Large Data Transactions in CoAP Networks, Sensors 14 (8) (2014) 15610–15638. doi:10.3390/s140815610.

[28] A. Lahmadi, A. Boeglin, O. Festor, Efficient Distributed Monitoring in 6LoWPAN Networks, in: CNSM - 9th International Conference on Network and Service Management - 2013, University of Zürich, Zurich, Switzerland, 2013.

[29] G. Gaillard, D. Barthel, F. Theoleyre, F. Valois, Monitoring KPIs in synchronized FTDMA multi-hop wireless networks, in: 2016 Wireless Days (WD), 2016, pp. 1–6. doi:10.1109/WD.2016.7461516.

[30] A. Karaagac, E. De Poorter, J. Hoebeke, In-Band Network Telemetry in Industrial Wireless Sensor Networks, IEEE Transactions on Network and Service Management 17 (1) (2020) 517–531. doi:10.1109/TNSM.2019.2949509.

[31] A. Karaagac, E. De Poorter, J. Hoebeke, Alternate Marking-based Network Telemetry for Industrial WSNs, in: 2020 16th IEEE International Conference on Factory Communication Systems (WFCS), 2020, pp. 1–8. doi:10.1109/WFCS47810.2020.9114490.

[32] X. Vilajosana, T. Watteyne, T. Chang, M. Vučinić, S. Duquennoy, P. Thubert, IETF 6TiSCH: A Tutorial, IEEE Communications Surveys Tutorials 22 (1) (2020) 595–615. doi:10.1109/COMST.2019.2939407.